Discrete and Hybrid Methods for the Diagnosis of Distributed Systems

Priscilla Fong Line Kan John

A thesis submitted for the degree of
Doctor of Philosophy
of
The Australian National University

Australian National University

College of Engineering and Computer Science
The Australian National University

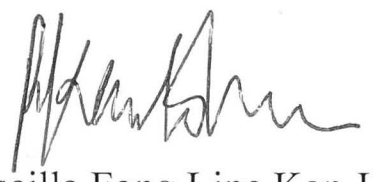
13 February 2012

Dedication

This thesis is dedicated to my family (my mother Caroline, my father Pakson, my two brothers Henri and Kenny) and my mentor Uncle Ho Wye.

Declaration

This thesis is an account of research undertaken between February 2008 and December 2011 at the Research School of Information Sciences and Engineering, College of Engineering and Computer Science, Australian National University, and at the Canberra Research Laboratory of the National Information and Communication Technology Australia (NICTA), both located in Canberra, Australia. Except where acknowledged, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

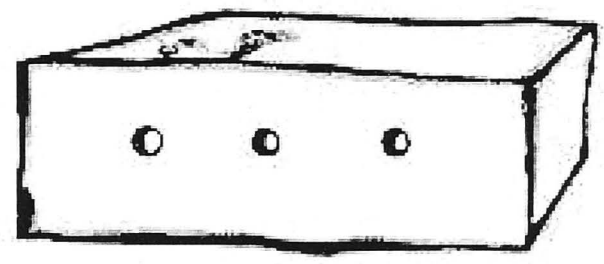


Priscilla Fong Line Kan John

13 February 2012

-Ça c'est la caisse. Le mouton que tu veux est dedans.*

(-This is the box. The sheep that you want is in there.)



The Little Prince, by Antoine de St Exupéry

The Samurai's Sword

The samurai clings to his sword as he slowly falls down. But nothing will allow that. His sword pushes him up to his feet. To leave the battlefield, he has no right. A power, so strong, stronger than the whole weight of the world emanates from his sword and gives him strength. He lifts his chin, prouder than ever. He knows he has to stand up no matter what. To look the enemy in his face and to win even before fighting, this is the attitude. To be a winner in advance and to pull victory by the hair is winning three quarters of the battle. The sword of the samurai, his most valuable asset, is his family. The love the family gives represents power, the strongest power that can be. With this power within you, fear not my friend. You may need to add willpower. I know you do not lack courage. Honour your family by coming out f the battlefield as undefeated as ever. Take your sword and lift your chin, our hearts are with you. [†]

[†]Prose for Uncle Ho Wye, 1997

Diagnosis *: from Greek diagnOsis, from diagignOskein to distinguish, from dia- + gig- nOskein to know. The art or act of identifying a disease from its signs and symptoms.* [‡]

[‡]from Merriam Webster Collegiate Dictionary - http://www.m-w.com

Abstract

Many important activities of modern society rely on the proper functioning of complex systems such as electricity networks, telecommunication networks, manufacturing plants and aircrafts. The supervision of such systems must include strong diagnosis capability to be able to effectively detect the occurrence of faults and ensure appropriate corrective measures can be taken in order to recover from the faults or prevent total failure. This thesis addresses issues in the diagnosis of large complex systems. Such systems are usually distributed in nature, *i.e.* they consist of many interconnected components each having their own local behaviour. These components interact together to produce an emergent global behaviour that is complex. As those systems increase in complexity and size, their diagnosis becomes increasingly challenging.

In the first part of this thesis, a method is proposed for diagnosis on distributed systems that avoids a monolithic global computation. The method, based on converting the graph of the system into a junction tree, takes into account the topology of the system in choosing how to merge local diagnoses on the components while still obtaining a globally consistent result. The method is shown to work well for systems with tree or near-tree structures. This method is further extended to handle systems with high clustering by selectively ignoring some connections that would still allow an accurate diagnosis to be obtained.

A hybrid system approach is explored in the second part of the thesis, where continuous dynamics information on the system is also retained to help better isolate or identify faults. A hybrid system framework is presented that models both continuous dynamics and discrete evolution in dynamical systems, based on detecting changes in the fundamental governing dynamics of the system rather than on residual estimation. This makes it possible to handle systems that might not be well characterised and where parameter drift is present. The discrete aspect of the hybrid system model is used to derive diagnosabilty conditions using indicator functions for the detection and isolation of multiple, arbitrary sequential or simultaneous events in hybrid dynamical networks.

Acknowledgements

Writing a PhD thesis is a bit like running an academic marathon. It takes endurance, willpower and tremendous support. I could not have done it without support from the people mentioned below.

My deepest gratitude goes to my PhD supervisors for their invaluable help and support without which this thesis would not have been possible: Sylvie Thiébaux, Alban Grastien and Jussi Rintanen. To Sylvie, as Panel Chair, thank you infinitely for your guidance, for ensuring I stay sane and for pushing me to get to the goal. To Alban, as my main supervisor, I appreciate and value your guidance, the countless hours of discussions, the board drawings and the attention to details. To Jussi, as my advisory supervisor, I thank you for the discussions, the pointers and organising helpful reading groups.

I would also like to thank my collaborators Yannick Pencolé and Lachlan Blackhall for our fruitful work together.

I cannot forget my office mates, especially Debdeep, Cong, Oliver, Leon, Ayman and Cindy. We have had a lot of fun being in the same boat together. I will treasure memories of numerous take-away lapasa lunches with Debdeep and of cross-fit training with Leon. I would also like to thank the Friday Drinks troupe for making sure I have some social activities in my calendar.

To the girls from the Spellbound Bellydance Troupe, especially Ginnie and Ludi, I express my gratitude for having a great place and group to chill, exercise and giggle.

Special thanks to my friend Lucy for her continuous support and friendship over the last ten years. Your friendship has seen me through the emotional rollercoaster ride that was involved over the last four years. I thank you, Liang and Fara for understanding my anti-social behaviour induced by thesis obligations.

To my boss Mick, I would like to express appreciation for your faith in me; in giving me a job before I submit my PhD and supporting me to make sure I do write the document.

I also want to thank Matt from the Street Theatre Café for the wonderful coffees he makes that were crucial to firing up my brain, and for the broad smile he served it with.

To my extended family, my aunts, uncles and cousins, who believe in me and have always colluded to make me feel special. Extra special thanks to Uncle Ho Wye, who has been my mentor throughout my life, who has inspired me to love science, and who has pushed me to excel.

Last but not least, to my close family; my mum Caroline, my dad Pakson, my two brothers

Henri and Kenny, words are not enough to tell you how much you mean to me. Thank you for your unfailing faith and love, I love you too.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Diagnosis is historically one of the first topics to be tackled in Artificial Intelligence with the view to automation. Diagnosis is a form of reasoning and allows inference from existing and available knowledge to give a better understanding of what has happened on a system, especially when faults occur. Key activities of the modern world are increasingly reliant on the proper functioning of complex systems such as electricity networks, telecommunication networks, manufacturing plants and aircrafts. Diagnosis is critical in the supervision of such systems to ensure their smooth operation as it is essential to capture the occurrence of faults so that appropriate control actions can be taken by the supervisor to manage and/or recover from the faults or prevent total failure. Given observations on a physical system, diagnostic reasoning can determine if a system is in nominal operation mode or if something went wrong (*i.e.* a fault occurred). Two diagnostic subtasks [67] of interest can be pointed out:

1. *Fault detection*, which is concerned with determining *whether* a fault has occurred or not;

2. *Fault identification*, which is concerned with finding out *what* fault has occurred.

Evidently, fault detection is the most basic feature of any diagnostic engine. It detects discrepencies between system measurements and expected behaviour. Fault identification is a refinement on fault detection and reasons on the model and observation to figure out which fault(s) occurred.

Complex systems often have a distributed nature, *i.e* they consist of many interconnected components. While the behaviour of individual components of a system can be simple, the interaction between components gives rise to an emergent complex behaviour of the overall network. Hence as those systems increase in complexity and size, their diagnosis becomes increasingly challenging. Traditional methods used do not scale up nicely and are too computationally costly, if not impossible to apply.

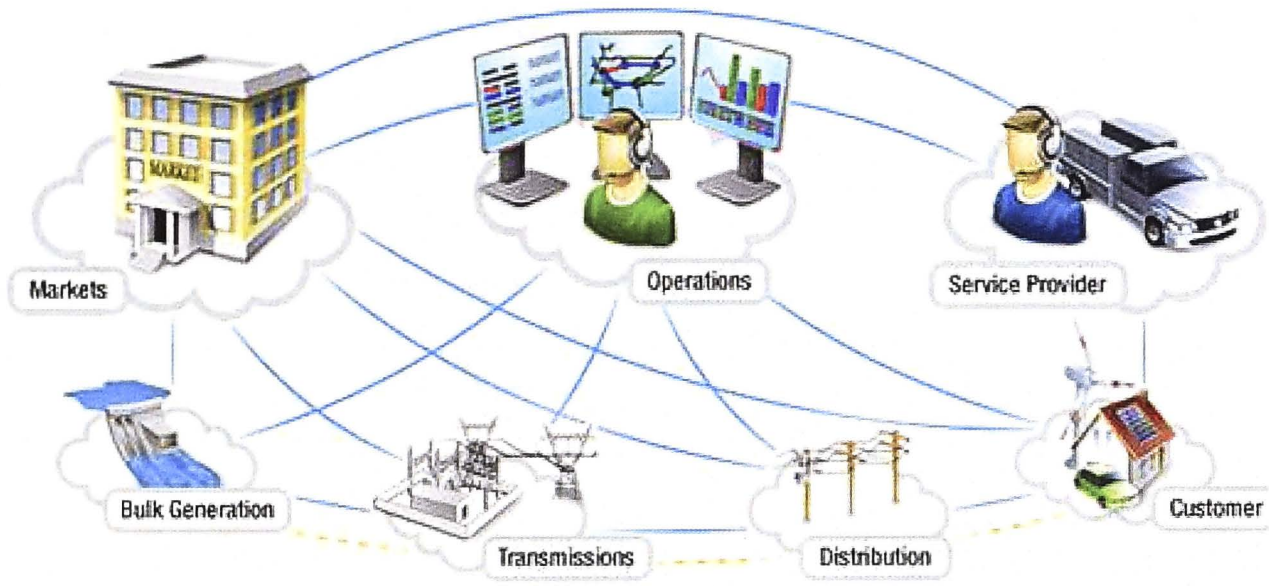


Figure 1.1: The National Institute of Standards and Technology (NIST) Smart Grid Conceptual Model, used with permission, © 2012 IEEE, obtained from [42].

1.1 Context

As complex systems become more pervasive, ways to manage them effectively need to be found. Automation of some, if not all, functions of these systems makes it possible to achieve greater efficiency in managing them while at the same time reducing the risk of errors and failures. Information and communication technology (ICT) plays an important role in developing appropriate methods and tools for making such systems as autonomous as possible. The Federal Government has recognised the need for research in ICT and established NICTA (National Information and Communication Technology Australia) in 2002 as part of the Backing Australia's Ability Initiative. The work presented in this thesis was done within the context of two projects at NICTA: **SuperCom** and its successor **AI for the Smart Grid**.

Supercom (*Model-based Supervision of Composite Systems*) targeted *composite* systems which are systems consisting of interconnected components exhibiting complex behaviour. The main focus of the project was on model-based (*i.e* an underlying model of the system is available) supervision with the goal of conferring upon systems the ability to self-diagnose problems and keep operating as efficiently as possible until faults are repaired.

The motivating application for the project **AI for the Smart Grid** is the supervision of electricity networks with intelligent behaviour. Traditional power grids, where electricity is distributed in one direction, from generators to consumers, do not meet the needs of a growing and resource-conscious society anymore. Existing infrastructure does not scale up easily to handle increased demand, is vulnerable to cyber-attacks, and does not cope well with injection of intermittent renewable power sources, such as wind and solar. Hence many countries are making the move towards a "smart" grid, which is a power grid overlayed with a communications layer that provides two-way data communication among all components of the grid so that information about the conditions of the various components can be used to better manage and control the grid. A conceptual model of the smart grid is shown in figure 1.1.

Whether it be electricity networks, or other types of interconnected systems, automated diagnosis is an important feature of their supervision. My work is aligned with the goal pursued within both **Supercom** and **AI for Smart Grid** of developing methodologies and tools that will allow such systems to detect the occurrence of faults and either recover from them or minimise their impact.

1.2 Thesis Aims

The main objective of this thesis is to contribute to the body of work on model-based diagnosis of distributed systems, with the main motivating application being electricity networks.

Distributed systems consist of many interconnected components, each having their own local behaviour, which interact together to produce an emergent global behaviour that is complex. Reasoning at the global level over the entire network is often very difficult, if not impossible, due to the complexity of required algorithms. Decentralised diagnosis methods ([83, 29, 96]) have therefore been developed to help in managing this complexity by avoiding a monolithic global computation. This is the approach on which we based ourselves for diagnosis on distributed systems using a discrete event system viewpoint. We propose in the first part of this thesis to extend existing work in the field to take into account the topology of the system in choosing how to merge the local diagnoses on the components or what connections can be ignored but still obtain a globally consistent result without having to explicitly calculate a global diagnosis.

Real world distributed systems are dynamic in nature, *i.e.* their evolution is given by a function over time. This behaviour, described by continuous variables (*e.g.* voltage, current or capacitance), is what is measured by sensors at the basic level. Many tools and techniques have been developed in the Control Theory field to diagnose and control systems by processing data provided in their continuous variables. However continuous models often have a level of granularity that is too high for effective logic-based reasoning on them. To address this issue, a Discrete Event System (DES) model [20] of the system can be abstracted to make it easier to reason on the system. In a DES model the evolution of the system is characterised by discrete state changes triggered by the occurrence of discrete events on the system. Performing this abstraction however means that some information is lost. There could be situations where the continuous dynamics information could help better isolate or identify faults. To get the best of both worlds, hybrid system approaches that unify both discrete and continuous techniques have been explored ([54, 72, 47]). We situate the work presented in the second part of this thesis along this direction. We build on existing work by taking in consideration situations where limited or no apriori knowledge of the system operarations and parameters exist, or there is parameter drift in the system. The traditional residual generation approach cannot be applied in these situations. Our hybrid system approach detects changes in the fundamental governing dynamics of the system as the onset for operational mode changes. Hence, both discrete and continuous techniques can be applied to a system in a hybrid framework, the combination of which provides powerful diagnosis tools.

1.3 Thesis Contribution

The research in this thesis has been done from two key perspectives, a discrete event system viewpoint and a dynamical system viewpoint. Taking a DES approach makes it possible to apply logic-based techniques for diagnosis. One main contribution of this thesis takes inspiration from graph theory and applies a junction tree transformation to the topology of the system for diagnostic reasoning on the system. This results in a model where each cluster of the resulting junction tree is a subsystem of the network (*i.e.* the entire system). The advantage of doing this is that local consistency ensures global consistency in a tree, and hence in a junction tree. Furthermore, the properties of a junction tree makes it possible to achieve global consistency by applying synchronisation among the clusters in two passes along the tree.

A junction tree approach works best for system that have an actual tree or near-tree topology. When this is not the case and the components have a large number of connections between them, diagnosis can still be highly complex due to the clusters having a large size. We address this problem by ignoring certain connections on the system, which reduces the cost (*i.e.*complexity) of the diagnosis. We handle the problem of the possible accompanying reduction in accuracy of the diagnosis by performing an off-line accuracy analysis to determine which connections can be ignored without compromising the accuracy of the diagnosis obtained.

While discrete techniques are powerful for reasoning at a higher abstract level, they are not sufficient by themselves for handling dynamic systems. To this effect, there is value in retaining information about the continuous operation of a dynamic system. We present a hybrid system framework that models both continuous dynamics and discrete evolution in dynamical systems based on detecting changes in the fundamental governing dynamics of the system rather than on residual estimation. This makes it possible to handle systems that might not be well characterised and where parameter drift is present.

We derive diagnosability conditions using indicator functions for the detection and isolation of multiple, arbitrary sequential or simultaneous events in hybrid dynamical networks, made possible with the discrete aspect of the hybrid model. While hybrid dynamical networks are our focus, the diagnosability results can easily be extended to any discrete event system model since it defined on the discrete part of the system.

1.4 Thesis Structure

The work presented in this thesis has been structured into two main parts. Results pertaining to a DES formalism are given in Part I and results pertaining to a Hybrid System formalism are given in Part II. As a prelude to both parts, a survey of the literature and important existing formalisms and definitions are provided in chapter 2. We visit the concept of Model-Based Diagnosis (MBD) and the various aspects of system modelling at different abstraction levels. We also present concepts relevant to dynamical systems and early efforts at probabilistic treatment. We recall the language

formalism widely used for representing operations on DES and the popular automaton modelling framework used for system description. Each chapter is mostly self-contained and where appropriate previously covered material is reproduced to facilitate reading.

Part I introduces diagnosis results on distributed systems obtained using discrete methods. Chapter 3 provides the necessary background for the work presented in the distributed discrete diagnosis area. It clarifies the notion of *diagnosis*, and associated concepts, from a discrete event system perspective, as relevant to this thesis. A proof that a globally consistent diagnosis is a solution to the diagnosis problem is also provided. To achieve global consistency without performing global computation is the focus of chapter 4, where a method is proposed for performing diagnosis on a distributed system that transforms the topological graph of the system into a junction tree first. In this way, clusters of subsystems are obtained. Clusters can be synchronised locally to achieve a globally consistent diagnosis. When systems are highly connected, the junction tree method is not as effective as for systems with tree or near-tree structure. We propose to ignore certain connections in chapter 5 to reduce the complexity of diagnosis on such systems. We avoid the reduction in accuracy that could happen by doing this by carrying out a preliminary off-line accuracy analysis to determine which connections can be safely ignored.

To further augment the diagnosis capability on distributed systems, we unify discrete diagnosis techniques with continuous diagnosis techniques within a hybrid system framework, which is the focus of part II of the thesis. Chapter 6 presents the context for our work in the hybrid system diagnosis area and the hybrid system framework relevant to our work. We detail the approach we take to diagnosis of hybrid dynamical networks in chapter 7, based on detecting changes in the fundamental governing dynamics of the system rather than residual estimation. We include the assumptions made and results on a small example network. Although a highly interconnected network exhibits behaviour that is complex, in chapter 8, we show that diagnosability criteria for such a network is only dependent on the number of events in the system and not on the number of components in the network. Sets of indicator functions are used to present diagnosability criteria.

We conclude by providing an overall assessment of the contribution of this thesis and outline directions for future work.

Chapter 2

Literature Review

Diagnosis can be viewed as the problem of finding out what happened on a system, especially when faults occur, given observations on the system. Research on finding an automated solution to this problem started in the 1940's with the foundation for *expert systems* being laid [79]. An *expert system* was viewed as a computer system that could emulate the decision-making ability of a human expert. The early expert systems were rule-based and used association rules between symptoms exhibited by a system and possible causes, similar to how a doctor would diagnose a patient. Associative (if-then) rules become harder to develop as systems increase in size. Moreover, some rules can become invalidated with even small changes to system configuration, and it has proven [32] quite difficult to isolate or update the invalidated rules. Another major limitation is that most often critical faults have low probability of occurrence and are hence hard to capture using associative rules until the faults are encountered, which could have disastrous consequences. Limitations of the early rule-based approaches led to the evolution of model-based approaches in the late seventies.

2.1 Model-based diagnosis (MBD)

In a model-based approach, a description of the system's behaviour, usually developed using mathematical means, called the system model is available. This model defines at least the normal behaviour of the system. If observations emitted by the system do not match the expected normal behaviour as encoded by the system model, then we know that a fault or faults have occurred (*fault detection*). Better diagnostic results can be achieved when predefined fault models exist as this makes it possible not only to tell that a fault has occurred, but also what fault has occurred (*fault identification*). MBDuses a declarative representation in which there is separation of knowledge representation (in a model) and reasoning (as diagnostic algorithms).

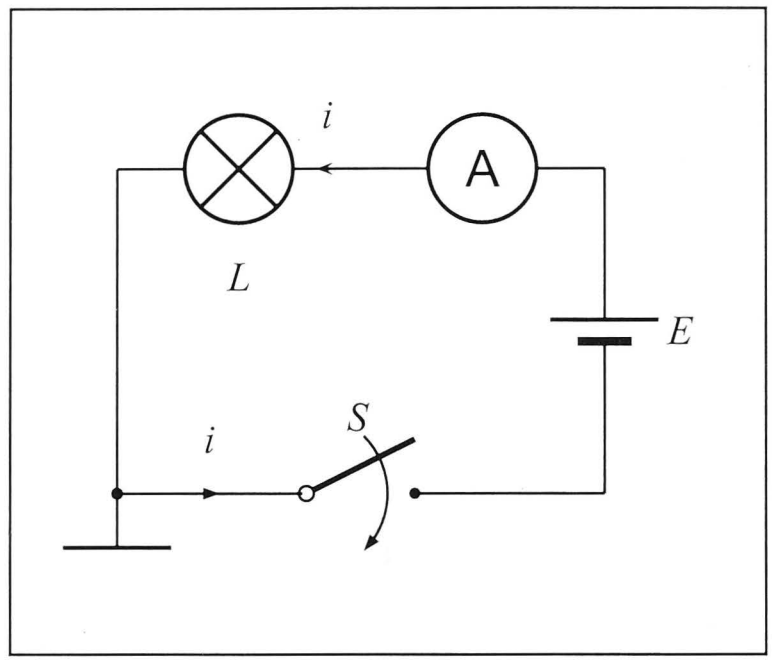


Figure 2.1: Example of electric circuits: with lamp and switch.

2.1.1 System Model and Knowledge Representation

A model is an abstraction of the real world and is a representation of knowledge possessed about the system. We define the following notions as pertain to any given model of a system.

Definition 1 (Model Correctness)**.** The *correctness* of a model is defined with respect to its input-output behaviour. A model is *correct* if for every possible input, it produces an output that matches the observations.

Definition 2 (Model Accuracy)**.** The *accuracy* of a model gauges how closely the model's predictions match the reality being modeled. It must be defined relative to some referent observations or to a more detailed base model.

Definition 3 (Model Resolution)**.** The *resolution* of a model refers to its level of refinement and is determined from the precision of its output. (*e.g.* A qualitative model has lower resolution than a quantitative model.)

We illustrate the notions introduced in definitions 1, 2 and 3, using two simple examples illustrated by figures 2.1 and 2.2.

In figure 2.1, we consider a simple circuit with a lamp L having a resistance of 10 Ω and a switch connected to a battery of 10 V. A basic model can be derived for the current and voltage across the lamp L with respect to the state of the switch S (table 2.1).

The model shown in table 2.1 gives us a relationship between the state of the switch and the value of current across the lamp. This model is *correct* for the circuit given in figure 2.1(a), assuming a battery of 10 V. For every value of the switch S, it predicts a value for the current i that matches the observations (as seen on the ammeter).

Switch S	**Voltage across lamp** L	**Current across lamp** L
ON	10 V	1 A
OFF	10 V	0 A

Table 2.1: Simple Model for lamp and switch circuit.

We now consider a variation to this model given in table 2.2 where a rheostat (variable resistor) is inserted in series with the lamp. Let the resistance of the rheostat be represented by R_v.

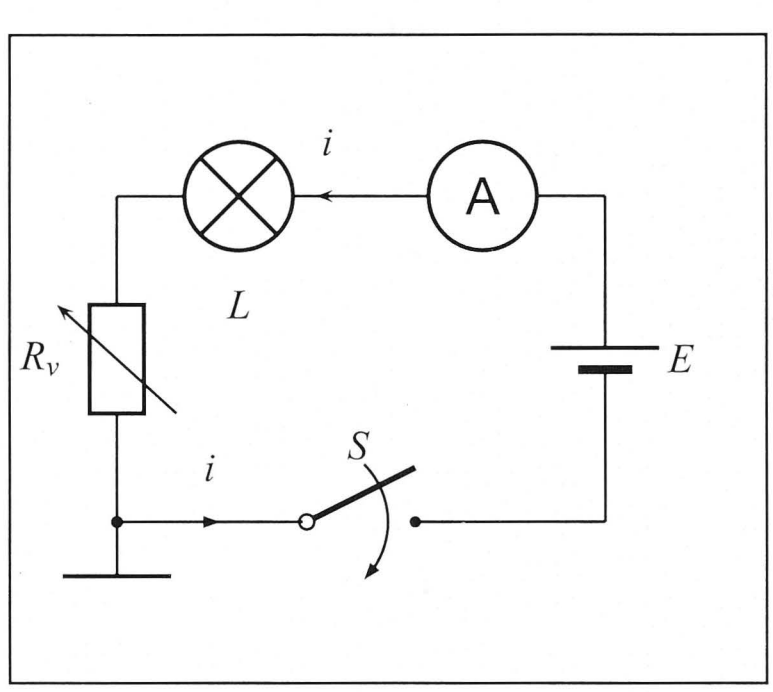


Figure 2.2: Example of electric circuits: with lamp, switch and rheostat.

Switch S	**Voltage across lamp** L **(V)**	**Current across lamp** L **(A)**
ON	$5 \leq V_L \leq 10$	$0.5 \leq i \leq 1$
OFF	10	0 A

Table 2.2: Simple model for lamp and switch circuit with rheostat in series.

Model 2.1 is no longer *correct* for the circuit given in figure 2.2 because the value of current it predicts does not match the observations on current anymore when the switch is ON and the value of R_v is non-zero. We can adjust the model to take into account the effect of the rheostat as described in table 2.2. Model 2.2 is *correct* for the circuit given in figure 2.2.

Although correct, model 2.2 only broadly represents changes with the resistance values (R_v) of the rheostat. We can isolate the case where $R_v = 0\Omega$ as it is equivalent to having no rheostat connected. This allows us to narrow down the value for current and voltage for that particular case. The given model is shown in table 2.3. Model 2.3 is more *accurate* than model 2.2 for the case where $R_v = 0\Omega$ because the values it predicts for current and voltage, given$R_v = 0\Omega$, is closer to

Switch S	**Voltage across lamp** L **(V)**		**Current across lamp** L **(A)**	
	$R_v = 0\ (\Omega)$	$0 < R_v \leq 10\ (\Omega)$	$R_v = 0\ (\Omega)$	$0 < R_v \leq 10\ (\Omega)$
ON	10	$5 < V_L < 10$	1	$0.5 < i < 1$
OFF	10	10	0	0

Table 2.3: Higher accuracy model for lamp and switch circuit with rheostat in series.

Switch S	**Voltage across lamp** L **(V)**	**Current across lamp** L **(A)**
ON	iR_L	$\frac{10}{R_v + R_L}$
OFF	10	0 A

Table 2.4: Refined model for lamp and switch circuit with rheostat in series.

their true values. Model 2.3 also has higher *resolution* than model 2.2 in the case $R_v = 0\Omega$ because it narrows down the range of possible values that is predicted by the latter model. Both models are equally accurate and have the same resolution for the case where $0 < R_v$ lg 10Ω since they both predict the same range of values for current and voltage.

Model 2.3 is only able to estimate the value of current i to be within a certain range (0.5-1 A) for values of R_v that are not zero. It might be necessary to get a better idea of the value of current i and in that case the model described in table 2.4 would be more appropriate. This model is able to predict more closely the value of current i for all values of R_v, including cases where $0 \leq R_v \leq 10$. Model 2.4 is therefore more *accurate* and has higher resolution than both model 2.2 and 2.3.

The *correctness*, *resolution* and *accuracy* of a model impact the results obtained from reasoning done using the model. In this thesis, we assume that models used are *correct*. The resolution of the model used and its accuracy with respect to given variables are often determined by the formalism used for modeling and the abstraction level at which the modeling is done. The general modeling formalism within the FDI community is the use of differential equations to represent system evolution (as described in section 2.2.1). In the DX community, logic based representations are used (some of these will be mentioned and described in section 2.2.2). Examples include basic logic predicates [71], compact logic structures such as binary decision diagrams (BDD) [93], and more expressive structures like Petri nets and automata [20]. Automata, because they are the basis of the chosen modeling formalism in this thesis, will be further described in section 2.2.5.2.

Abstraction levels are explained in the next section.

2.1.2 Abstraction levels

When choosing a model to use for MBD, the first step is to decide what information to include into it. This step has serious consequences on the quality of the diagnosis since any information that is discarded at this stage would then be inaccessible during diagnosis. However, it is unrealistic

to describe the system down to its finest details, especially if it is a large system, as this would make the diagnosis task intractable. One way to handle this issue is to trade-off what information to keep. Another way is to use multiple abstraction levels within the model organised through a hierarchy (functional abstraction) to support diagnosis reasoning task. The latter way is specially useful to handle systems whose available knowledge is heterogeneous (*i.e.* both continuous and discrete). We distinguish four epistemological classes within a functional abstraction hierarchy:

1. **Structural Knowledge:** This is the lowest level of abstraction and captures knowledge about system topology. It contains description of the components making up the system and their connections.

2. **Behavioural Knowledge:** This class captures the physical laws underlying the behaviour of components that make up the system.

3. **Functional Knowledge:** This class describes the roles of components in the processes that happen in the system.

4. **Teleological Knowledge:** This is the highest level of abstraction and describes the goals of the system and the conditions required to fulfill these goals. It is totally detached from system implementation details.

2.2 Different Approaches to MBD

As mentioned previously, the two distinct communities, FDI and DX, have developed their own approaches to MBD. These approaches exhibit diversity in the modeling details but similarity in the overall reasoning. The modeling of a system can be classified in two distinct categories depending on whether the emphasis for describing system evolution is on the system state (state-based systems) or on events occuring in the system (event-based systems). State-based modeling is used within both FDI and DX communities whereas event-based modeling is more prevalent in DX. Some diagnosis work has also started looking at using both a state-based and event-based emphasis to address hybrid systems (systems which take both discrete and continuous evolution into account).

2.2.1 FDI Approach to MBD

The FDI approach inherits techniques from automatic control and generally looks at the modeling of continuous variables. Generally, a continuous state space model is adopted. We explain the concept of *state space* as used in the FDI community in this section.

2.2.1.1 System Model in FDI

We present here an overview of system modelling concepts as used in the control theory community to clarify their meaning and to compare later with the approach used in the computer science

community. The concepts and definitions presented are standard in the control theory community and are covered in textbooks such as [54, 20].

We consider the model of a system to be a means to relate input to the system to the output. The input to the system is a set of measurable variables that can be varied over time, written as a vector $\mathbf{u}(t)$. The output to the system is another set of selected variables which can also be directly measured while varying $\mathbf{u}(t)$. We denote the set of output variables as a vector $\mathbf{y}(t)$. We thus have an input-output relationship for a system where the model provides the link between input and output, as illustrated in figure 2.3.

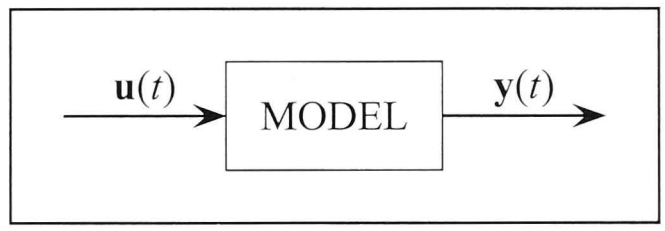


Figure 2.3: System Model showing the relationship between input and output

Expanding on the idea of an input-output relationship of a system as presented above, we introduce the concept of a system *state*.

Definition 4. (System State) The *state* of a system at a time instant t describes the condition of the system at that point in time. State equations (or functions) describe what is happening with the input $\mathbf{u}(t)$ to obtain the output $\mathbf{y}(t)$ while the system is in a state represented by the set of state variables in a vector $\mathbf{x}(t)$. The complete state space model of a system is as given in definition 6.

State equations specify the state $\mathbf{x}(t)$ of a system for all $t \geq t_0$ given $\mathbf{x}(t_0)$, system parameters θ and the input function $\mathbf{u}(t), t > t_0$. System parameters are intrinsic to the system being modeled and are illustrated in example 2.2.1. State equations can take different forms and are generally accompanied by an output equation to give a complete system description. Most often, in control theory, they are given in terms of differential equations of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \theta) \tag{2.1}$$

For convenience, we drop the t in bracket in equation 2.1 such that $\mathbf{u}$ is the same as $\mathbf{u}(t)$ and similarly for $\mathbf{x}$ and $\mathbf{x}(t)$. $\dot{\mathbf{x}}$ is the derivative of $\mathbf{x}$ (with respect to time). f is a set of continuous (hence time dependent) functions that describe the dynamics of the system. This leads us to the definition of a dynamical system.

Definition 5. (Dynamical System) A system that is *dynamical* is a system whose evolution can be described by a set of differential equations that gives the state of the system in a small future time step, as illustrated by equation 2.1.

The *state space* of a system is the set of all possible values that the state may take.

Definition 6. (State Space Model) A *State Space Model* consists of a set of state and output equations that completely describes the dynamic operation of a system. It is given as:

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}, \theta) \\ \mathbf{y} &= h(\mathbf{x}) \end{aligned} \tag{2.2}$$

$\dot{\mathbf{x}}$ is the derivative of the state vector $\mathbf{x}$. Equation 2.2 completely describes the behaviour of a dynamical system and is the underlying system description used in the control theory community. It is often referred to as the *state space representation* of a system.

If the system can be solved (usually by integration), then starting from an initial point, it is possible to determine all its future points. This ensemble of points is known as a *trajectory* in the continuous dynamical system literature.

Definition 7. (FDI Trajectory) A *trajectory*, in an FDI context, is a sequence of values of the system state $\mathbf{x}(t)$ from an initial time step t_0 to a final time step t_f such that $t_0 \leq t \leq t_f$. We will refer to it as a *state trajectory*.

In this subsequent sections we will use the term *state trajectory* to refer to a trajectory in the FDI sense to distinguish it from the meaning of the term 'trajectory' in the computer science literature (which will be clarified in section 2.2.5.2).

Example 2.2.1. (Spring-Mass System)

We use a system example from [20] to illustrate the terms and concepts used to describe the continuous part of a hybrid system. Consider a spring-mass system as shown in figure 2.4. $y(t)$ is the output of the system and is a measurement of the displacement of the mass m at time $t, t > 0$. At time $t = 0$, the mass is displaced from its rest position by an amount $u(0) = u_0 > 0$ and released. The motion of the mass can be defined by a harmonic oscillation described by the second-order differential equation 2.3, with initial conditions $y(0) = u_0, \dot{y}(0) = 0$.

$$m\ddot{y} = -ky \tag{2.3}$$

In this simple system, we could assume we want to control the initial displacement $u(0)$ and observe the position of the mass as a function of time. We thus have a model where the input is the function $u(t)$ given by

$$u(t) = \begin{cases} u_0 & t = 0 \\ 0 & otherwise. \end{cases} \tag{2.4}$$

The output $y(t)$ is the solution of the differential equation 2.3 and we assume that k and m are constant.

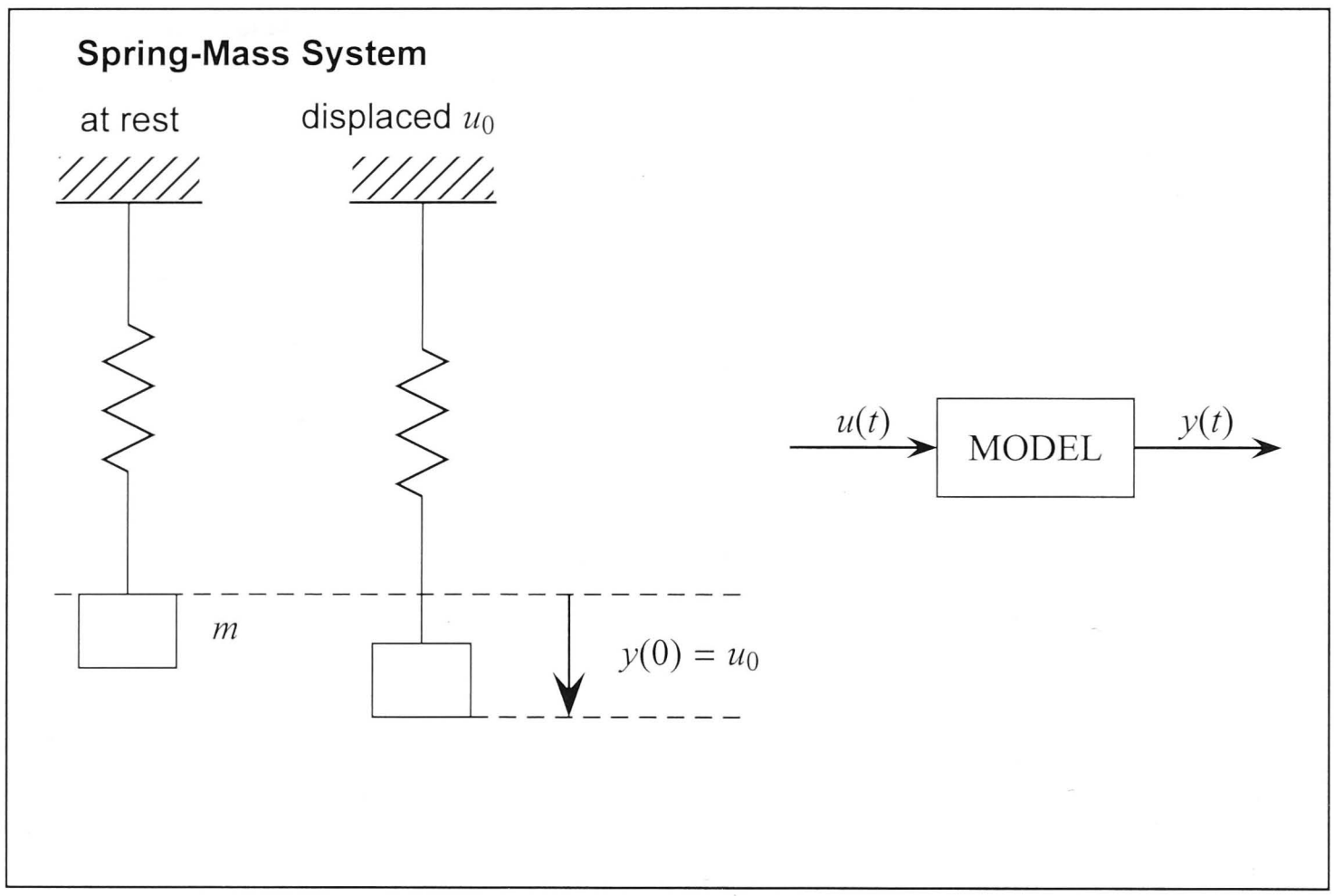


Figure 2.4: Spring-Mass System Example

We want to show how this problem is cast into a state space representation, *i.e.* in the form of equation 2.2

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \theta)$$
$$\mathbf{y} = h(\mathbf{x})$$

First, we define the mass displacement as a state variable and denote it by $x_1(t)$. This allows us to write a trivial output equation

$$y(t) = x_1(t) \tag{2.5}$$

We can thus rewrite equation 2.3 as

$$m\ddot{x_1} = -kx_1 \tag{2.6}$$

Equation 2.6 can be rewritten as a first order differential equation, so we introduce an additional state variable $x_2(t)$ defined as $\dot{x_1} = x_2$. Hence, we can rewrite equation 2.6 as

$$\dot{x_2} = -\frac{k}{m}x_1 \; ; \; \dot{x_1} = x_2 \tag{2.7}$$

This now gives us a complete model with two state variables and one output variable.
In this example system, the system parameters θ are given by the two constants k and m.

In simple systems like the one just presented, the model can be easily deduced and the system parameters are assumed known. However, in complex systems, often the model dynamics are not known and need to be determined in a process called *system identification*. The parameters included in these equations are never known with absolute precision and need to be estimated. There is a whole area of work on estimating the values of system parameters. This area draws from statistics and signal processing and is known as *estimation theory* [69, 4].

2.2.1.2 FDI Methodologies

Residuals A variety of methodologies have been developed in the control theory field for system diagnosis. One very popular approach is the parity space approach which uses techniques involving residuals. A residual is a function $\mathcal{F}(\hat{\mathbf{y}} - \mathbf{y})$ that is dependent on the difference between the nominal ($\mathbf{y}$) and observed ($\hat{\mathbf{y}}$) system output. Deviations of the observed state trajectory from some nominal trajectory are indicative of the occurrence of faults. Residual functions are designed carefully to respond to the presence of certain faults but are generally sensitive to noise. Parameter values are also assumed to be known and not to drift (*i.e.* not to undergo deviations). Residual techniques are well covered in [82, 50, 44].

Parameter Estimation Parameter estimation is a statistical technique that uses measurements on the system to find approximations to parameters in a model (*e.g.* k and m in the example presented above) that cannot be determined directly. This can help in detecting faults that manifest as parameter changes. It can also be used to complement a residual framework and make it more robust to noise. A variety of powerful methods have been developed in the field of estimation theory including least square methods, Kalman filters and particle filters [58, 23].

2.2.2 DX Approach to MBD

The DX approach originated from computer science and generally applies to discrete systems. The emphasis is more on the qualitative aspect of the behaviour of the system, rather than the precise evolution of its continuous variables over time. Using the mass-spring example presented, while a state space model tells us the exact value of the displacement of the spring at any given point in time, a qualitative discrete model might only contain the abstracted information of whether the mass is at rest or moving. This is in line with the notion of qualitative physics as presented in [63]. The motivation for a more abstract qualitative approach is to focus on the core knowledge that underlies human intuition (as what is pointed to by cues and notions of 'intelligent' thinking in the fields of psychology, education and artificial intelligence). Humans appear to use a qualitative causal calculus in reasoning about the behaviour of their physical environment. Hence using a qualitative model that abstracts the relevant information helps in developing automated diagnostic reasoning algorithms for systems.

The modeling of these discrete systems can be either state-based or event-based. The original approaches are state-based and looked at static systems, with first-order logic being used to model the system [85]. Subsequent state-based frameworks, despite being formulated in other logical frameworks (*e.g.* propositional logic or constraint networks), inherit a similar system model definition.

2.2.3 DX State-based System Model

A state-based system model in DX is considered to consist of the following sets:

- *COMPS* is the set of system components.
- *SD* is the system description written as a set of propositions in first order logic. Faulty components are designated by a unary predicate $AB(\ )$, taken to mean 'abnormal'.
- *OBS* is the set of observations, also written as first order logic propositions.

When $c \in COMPS$, the clauses $AB(\ )$ and $\neg AB(\ )$ are called *AB*-literals.

We note that the notion of time is not included in the description. The elements of *COMPS* are involved in defining *SD*. Although the focus is often on the component(s) that is(are) faulty, and hence the emphasis on *COMPS*, this need not always be the case, *i.e. COMPS* could very well be replaced by a set of variables of interest.

SD gives the relations between inputs, *COMPS*, unobservable variables and outputs of the system. Inputs and outputs of the system are observable and form part of *OBS*.

2.2.3.1 Conflict-based diagnosis

A diagnosis can be obtained from the system model presented above by a conflict-based approach, an approach first introduced by [85]. We give a modified definition as found in [40].

We first define the notion of *satisfiability*.

Definition 8. (Satisfiability) Given a logical expression E, we say that E is *satisfiable* if there exists some assignment to the variables in E such that E logically evaluates true.

$$E \not\models \bot \text{(satisfiable)}$$
$$E \models \bot \text{(not satisfiable)}$$

We now define the notion of *conflict* where a conflict describes a situation where a diagnosis hypothesis is not satisfiable.

Definition 9 (Conflict)**.** Given a state-based system model, we say that a conflict arises when a diagnosis hypothesis Δ is not consistent with respect to the system model and observations received. In such a case we can write: $SD,\ OBS,\ \{\neg AB(c) \mid c \in COMPS \setminus \Delta\} \models \bot$

Basically, this means that if the observations on the system conflict with the way the system is expected to behave (as encoded in the system model), then we have a diagnostic problem at hand. The diagnostic task consists of determining which system component(s) is(are) behaving abnormally.

This now allows us to define diagnosis, reformulated as in [84], as follows:

Definition 10. (DX Diagnosis) Let $C_1 \subseteq COMPS$, $C_2 \subseteq COMPS$ be two sets of components. We define $\mathcal{D}(C_1, C_2)$ as the conjunction

$$\mathcal{D}(C_1, C_2) = \left(\bigwedge_{c \in C_1} AB(c) \right) \wedge \left(\bigwedge_{c \in C_2} \neg AB(c) \right)$$

Let $\Delta \subseteq COMPS$ be a set of suspected abnormal components. The proposition $\mathcal{D}(\Delta, COMPS \backslash \Delta)$ is a diagnosis for $(SD, COMPS, OBS)$ if and only if the following proposition is satisfiable:

$$SD \cup OBS \cup \mathcal{D}(\Delta, COMPS \backslash \Delta)$$

The case where $\Delta = \emptyset$ is trivially the nominal operation case. There can be more than one candidate set Δ that would result in the above proposition being satisfiable. The different possibilities for Δ form the set of hypotheses for the diagnosis $\mathcal{D}(\Delta, COMPS \backslash \Delta)$ of the system. When a fault is present, the behaviour of the faulty component is unspecified. The diagnostic task then consists in identifying conflicts, *i.e.* finding sets of components that cannot all be in normal behaviour mode, and establishing each diagnostic hypothesis as a hitting set of all conflicts.

When faulty behaviour is unspecified, it is possible to suspect every component to be faulty. This implies that the conjunction $\mathcal{D}(COMPS, \emptyset)$ is always a diagnostic hypothesis in all situations. In general, one expects an exponential amount of diagnostic hypotheses with respect to the number of components. It is therefore necessary to narrow down the set of diagnostic hypotheses (which we also refer to as *diagnoses*). One popular method is to consider only minimal conflict sets and their minimal hitting sets in order to produce minimal diagnoses [45, 85].

Definition 11. (Minimal Diagnosis) A diagnosis $\mathcal{D}(\Delta, COMPS \backslash \Delta)$ is a *minimal diagnosis* if and only if for every $\Delta' \subset \Delta$, $\mathcal{D}(\Delta', COMPS \backslash \Delta')$ is not a diagnosis.

A minimal diagnosis is the most optimistic scenario as it makes the assumption that as few components as possible have failed. It is also usually the most probable.

2.2.4 Early Probabilistic Diagnosis treatment

An alternative method to minimal diagnosis to narrow down diagnoses is the use of probabilities. An early attempt is presented in [52] which we will give an overview of here.

We assume that to reduce the set of possible diagnosis candidates, the diagnostician has to perform measurements and calculations which will allow a differentiation to be made. We also assume

that those measurements are feasible on the system. A probability-based method for choosing the next measurement which best distinguishes the candidates is used. Let x_i be a measurable quantity that could reduce the diagnosis candidate set. Let v_{ik} be a possible value for x_i. Three sets are defined

1. R_{ik}: the set of candidates that would remain if x_i were measured to be v_{ik} (called the remaining set)

2. S_{ik}: the set of candidates in which x_i must be v_{ik} (called the selected set)

3. U_i: the set of candidates which do not predict a value for x_i (called the uncommitted set)

The set R_{ik} is covered by the sets S_{ik} and U_i: $R_{ik} = S_{ik} \cup U_i$, $S_{ik} \cap U_i = \emptyset$.

An evaluation function based on entropy is defined to determine how difficult (*i.e.* how many additional measurements are needed) it is to identify the actual candidate for each possible outcome of a measurement:

$$H = -\sum p_i \log p_i$$

H is the entropy of the candidate probabilities and p_i is the probability that candidate C_i is the actual candidate given the hypothesised measurement outcome.

Entropy is a good cost function because of some of its important properties. If every candidate is equally likely, we cannot discriminate between them, *i.e.* H is at a maximum. H approaches a minimum as one candidate becomes more likely than the others. The cost of identifying a candidate of probability p_i is proportional to $\log p_i^{-1}$ (*cf* binary search through p_i^{-1} objects). The expected cost of identifying the actual candidate is proportional to the sum of the product of the probability of each candidate being the actual candidate and the cost of identifying that candidate, *i.e.* $\sum p_i \log p_i^{-1} = -\sum p_i \log p_i$.

Unlikely candidates are expensive to find but because they occur rarely they only contribute little to the cost:

$$p_i \log p_i^{-1} \rightarrow 0 \text{ as } p_i \rightarrow 0.$$

Similarly, likely candidates. although they occur frequently, are easy to find, and also contribute little to cost:

$$p_i \log p_i^{-1} \rightarrow 0 \text{ as } p_i \rightarrow 1.$$

Candidates between those two extremes are more expensive to locate because they occur with significant frequency and hence the cost of finding them is significant.

Under the assumption that every measurement is of equal cost, the diagnostic objective is to identify the actual candidate in a minimum number of measurements. The best measurement is one which minimises the expected entropy of candidate probabilities resulting from the measurement.

The initial probabilities of candidates are computed from the initial probabilities of component failure (obtained from the manufacturer or by observation). It is also assumed that components

fail independently. The initial probability that a candidate C_i is the actual candidate C_a is given by:

$$p_i = \prod_{c \in C_i} p(c \in C_a) \prod_{c \notin C_i} (1 - p(c \in C_a))$$

Conditional probability of a candidate Given the measurement $x_i = v_{ik}$, Bayes' rule is used to compute the probability of a candidate C_l:

$$p(C_l \mid x_i = v_{ik}) = \frac{p(x_i = v_{ik} \mid C_l)\, p(C_l)}{p(x_i = v_{ik})} \tag{2.8}$$

There are three cases for evaluating $p(x_i = v_{ik} \mid C_l)$. If C_l predicts $x_i = w_{ik}$ where $w_{ik} \neq v_{ik}$, then the conditional probability is 0. If $w_{ik} = v_{ik}$, then the conditional probability is 1. Otherwise C_l predicts no value for x_i. It is assumed that there are m possible values for x_i and each of them is equally likely, giving a conditional probability of $1/m$.

$$p(x_i = v_{ik} \mid C_l) = \begin{cases} 0 & \text{if } C_l \notin R_{ik} \\ 1 & \text{if } C_l \in S_{ik} \\ 1/m & \text{if } C_l \in U_i \end{cases}$$

Substituting into Bayes' rule (equation 2.8) we obtain:

$$p(C_l \mid x_i = v_{ik}) = \begin{cases} 0 & \text{if } C_l \notin R_{ik} \\ \frac{p(C_l)}{p(x_i = v_{ik})} & \text{if } C_l \in S_{ik} \\ \frac{p(C_l)/m}{p(x_i = v_{ik})} & \text{if } C_l \in U_i \end{cases}$$

where $p(x_i = v_{ik}) = p(S_{ik}) + p(U_i)/m$.

2.2.4.1 Fault Models

Using a system model where only the nominal behaviour is specified, one can *detect* faults. It is also possible sometimes to *identify* the faults using information about the structure of the system. However, the obtained *fault identification* is often not precise enough resulting in a set of diagnoses that is too large to be useful enough. A solution to this problem is to use fault models. Moreover, some components may fail in different ways which need to be distinguishable for repair actions; *e.g.* an electrical component could behave like an open circuit or a closed circuit depending on the fault that occurred in it. The need and relevance of using fault models is well covered in [45].

2.2.5 DX Event-based Approaches

In the DX state-based approach, time is not generally explicitly handled when describing the model nor in performing the diagnosis task. While this is acceptable for static systems like electronic circuits, for dynamic systems we need a way to model the dynamics of the system. Work such as [33] and [95] deal with the issue by considering the value of derivatives or sequences of states (behaviours).

Another way to deal with dynamics in a system is to use an event-based approach, also known as the Discrete Event System (DES) framework, for which a rich body of work has been developed [20, 91]. In a discrete event system, system evolution is represented by the occurrence of discrete *events*. These events correspond to a change in the behaviour of the system including, but not limited to, the occurrence of faults.

The occurrence of an event can cause a change in the *state* of the system. The concept of a system state in the DX context, while similar to the concept of system state used in FDI (as given in section 2.2.1), still differs from the latter due to the difference in emphasis in the two approaches. In the DX context, the variables that describe the system state are discrete whereas in the FDI context, they are continuous. Continuous variables describing the quantitative continuous dynamics of a system (*e.g.* flow equations denoting the functioning of a water pump) can be abstracted to qualitative discrete variables that represents its qualitative behaviour (*e.g.* the pump could be *ok* or *blocked*) on which it is easier to perform logic-based reasoning [70].

2.2.5.1 Language Formalism

The evolution of Discrete Event Systems can be described using a language formalism which provides a universal description that is independent of the chosen implementation framework (*e.g.* automata or Petri nets, which will be explained in subsequent sections). The definitions (as given in standard textbooks such as [48, 20]) in this section are important as they act as preliminaries to the work presented in chapters 4 and 5.

Let Σ be a finite set of symbols. We call Σ an *alphabet*. We denote by $\Sigma^\star$ the set of all finite sequences on Σ; an element $\sigma = e_1.\cdots.e_n \in \Sigma^\star$ is called a *word* over the alphabet Σ; the empty word is denoted ε. A *language* $\mathcal{L}$ over Σ is a subset of $\Sigma^\star$.

Operations over languages include classical set operations, *e.g.* union, intersection, difference, power set, concatenation. In addition, two important operations on languages are *projection* and *synchronous product*, whose definitions are given below.

Definition 12 (Projection). The projection on Σ' of a word σ over $\Sigma \supseteq \Sigma'$ denoted $P_{\Sigma\to\Sigma'}(\sigma)$ keeps all the elements of σ in Σ'. Formally,

$$P_{\Sigma\to\Sigma'}(\sigma) = \begin{cases} \varepsilon & \text{if } \sigma = \varepsilon \\ P_{\Sigma\to\Sigma'}(\sigma') & \text{if } \sigma = e.\sigma' \text{ and } e \in \Sigma \setminus \Sigma' \\ e.P_{\Sigma\to\Sigma'}(\sigma') & \text{if } \sigma = e.\sigma' \text{ and } e \in \Sigma' \end{cases}$$

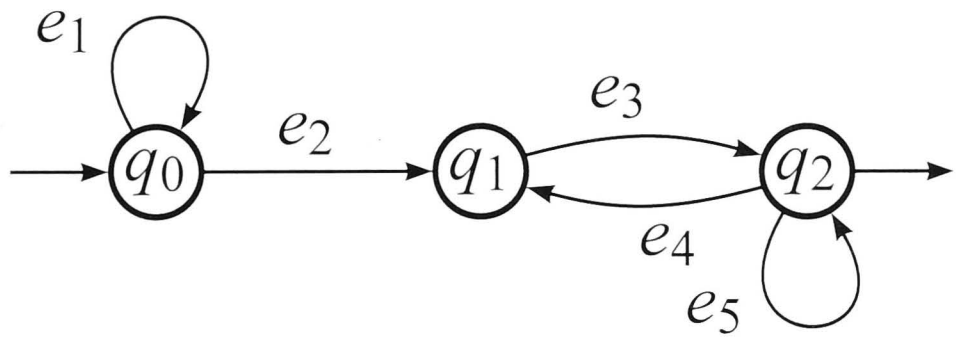


Figure 2.5: Automaton example

The projection on Σ' of a language $\mathcal{L}$ over Σ is denoted $P_{\Sigma\to\Sigma'}(\mathcal{L})$ and defined by $\{P_{\Sigma\to\Sigma'}(\sigma) \mid \sigma \in \mathcal{L}\}$. The inverse operation $P^{-1}_{\Sigma\to\Sigma'}$ of the projection from Σ to Σ' generates all the finite words over Σ whose projection on Σ' is the parameter: $P^{-1}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\sigma \in \Sigma^\star \mid P_{\Sigma\to\Sigma'}(\sigma) \in \mathcal{L}\}$.

Definition 13 (Synchronous Product)**.** The synchronous product $\otimes$ between two languages $\mathcal{L}_1$ over Σ_1 and $\mathcal{L}_2$ over Σ_2 computes all the words over $\Sigma_1 \cup \Sigma_2$ whose projection on Σ_i is $\mathcal{L}_i$: $\mathcal{L}_1 \otimes \mathcal{L}_2 = \{\sigma \in (\Sigma_1 \cup \Sigma_2)^\star \mid \forall i \in \{1,2\},\ P_{\Sigma_1\cup\Sigma_2\to\Sigma_i}(\sigma) \in \mathcal{L}_i\}$.

To consider system evolution in practical applications, languages used must be accepted by finite state machines, *i.e.* the language must be regular.

Definition 14 (Regular Language)**.** The collection of regular languages over an alphabet Σ is defined recursively as follows:

- The empty language $\emptyset$ is a regular language.
- The empty string language $\{\varepsilon\}$ is a regular language.
- For each $a \in \Sigma$, the singleton language $\{a\}$ is a regular language.
- If A and B are regular languages, then $A \cup B$ (union), $A \,.\, B$ (concatenation), and A^* (Kleene star) are regular languages.

Property 2.2.2 (Regular Language and Automata)**.** A language is regular if and only if there is a finite state automaton that accepts it.

This property leads us to the definition of a finite state automaton.

2.2.5.2 Automata

Definition 15 (Finite State Automaton)**.** A finite state automaton is defined by a tuple $\mathcal{A} = \langle Q, \Sigma, T, q_0, q_f \rangle$ where

- Q is the finite set of discrete states.

- Σ is the set of events.
- T is the set of transitions such that $T : Q \times \Sigma \rightarrow Q$. We write $(q_1, e, q_2) \in T$ to denote that the system evolves from state q_1 to q_2 upon the occurrence of the discrete event e.
- $q_0 \in Q$ is the initial state.
- $q_f \subseteq Q$ is the set of accepting or final states.

An example of a finite state automaton is shown in figure 2.5.

An automaton is said to be *deterministic* if for every event e from a given state, there is only one possible transition: $\forall q_i, q_a, q_b \in Q, \forall e \in \Sigma, \forall (q_i, e, q_a), (q_i, e, q_b) \in T, q_a = q_b$. The automaton $\mathcal{A} = \left\langle Q, \Sigma, T, q_0, q_f \right\rangle$ accepts the language $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^*$ defined by:

$$\mathcal{L}(\mathcal{A}) = \{e \in \Sigma^* \mid \exists q \in q_f, (q_0, e, q)\}$$

Given an automaton $\mathcal{A} = \left\langle Q, \Sigma, T, q_0, q_f \right\rangle$. Let $T^\star$ be the transitive closure of T. We can write the following:

$\forall$state $q, (q, \varepsilon, q) \in T^\star$.

$\forall$state $q, q', q'', \forall$sequence $\sigma, \forall$event e, if $(q, \sigma, q') \in T^\star$ and $(q', e, q'') \in T$, then $(q, \sigma, e, q'') \in T^\star$.

The language accepted by automaton $\mathcal{A}$ is defined by:

$$\mathcal{L}(\mathcal{A}) = \{\sigma \in \Sigma^* \mid (q_1, \sigma, q_2) \in T^\star, q_1 \in q_0, q_2 \in q_f\}$$

An automaton $\mathcal{A}$ is *live* if and only if for every state there is an outgoing transition: $\forall q_i \in Q, \exists e \in \Sigma, \exists q_j \in Q, (q_i, e, q_j) \in T$.

Equivalently, a language $\mathcal{L}$ is *live* if every word in $\mathcal{L}$ is a prefix of at least one other word in $\mathcal{L}$.

If every state in the automaton $\mathcal{A}$ can be an accepting (final) state, then $\mathcal{A}$ can be represented by the tuple $\langle Q, \Sigma, T, q_0 \rangle$. In this case, $\mathcal{L}(\mathcal{A})$ is prefix-closed, *i.e.* every prefix of every word in $\mathcal{L}(\mathcal{A})$ also belongs to $\mathcal{L}(\mathcal{A})$.

Definition 16 (DX Trajectory)**.** A trajectory, in the DX sense, refers to a path on an automaton $\mathcal{A}$ linking an initial state q_0 to a final state $q_k \in q_f$. We denote a trajectory by $\mathcal{T}(\mathcal{A})$ and define it formally as: $\mathcal{T}(\mathcal{A}) \subseteq Q, \mathcal{T}(\mathcal{A}) = \{q_1, \ldots, q_i, \ldots, q_k\}$ where $\forall i, 1 < i \leq k, \exists e \in \Sigma, (q_{i-1}, e, q_i)$.

The concept of a trajectory on an automaton distinguishes between event sequences that denote system behaviour (*i.e.* trajectories on the automaton) and event sequences that do not.

2.2.5.3 Petri Nets

Another possible implementation framework for DES is based on Petri nets, which are better suited to concurrent systems. A Petri net contains *places* and *transitions* that are usually connected by directed arcs. Places may contain *tokens* which can be interpreted as resources. Tokens may

move from one place to another by executing ('firing') actions. The arc linking a place to a transition has a corresponding *weight*. A transition is *enabled* if each of its input places contains as many tokens as the corresponding input arc weight indicates. When an enabled transition fires, each of its input places loses a number of tokens equal to the corresponding input arc weight, and each of its output places gains a number of tokens equal to the corresponding output arc weight. At each moment of its execution, the state of system modeled with Petri net is given by the current marking, *i.e.* the distribution of tokens in the places.

A Petri net $\mathcal{N}$ is defined as a tuple $\mathcal{N} = \langle P, T, F, W, M_0 \rangle$ where

- P is the finite set of places.
- T is the finite set of transitions.
- P and T are disjoint ($P \cap T = \emptyset$).
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation.
- $W : F \to (\mathbb{N} \backslash \{0\})$ is the arc weight mapping ($\mathbb{N}$ is the set of non-negative integers).
- $M_0 : P \to \mathbb{N}$ is the initial marking representing the initial distribution of tokens.

If $\langle p, t \rangle \in F$ for a transition t and a place p, then p is an input place of t. Likewise, if $\langle t, p \rangle \in F$ for a transition t and a place p, then p is an output place of t.

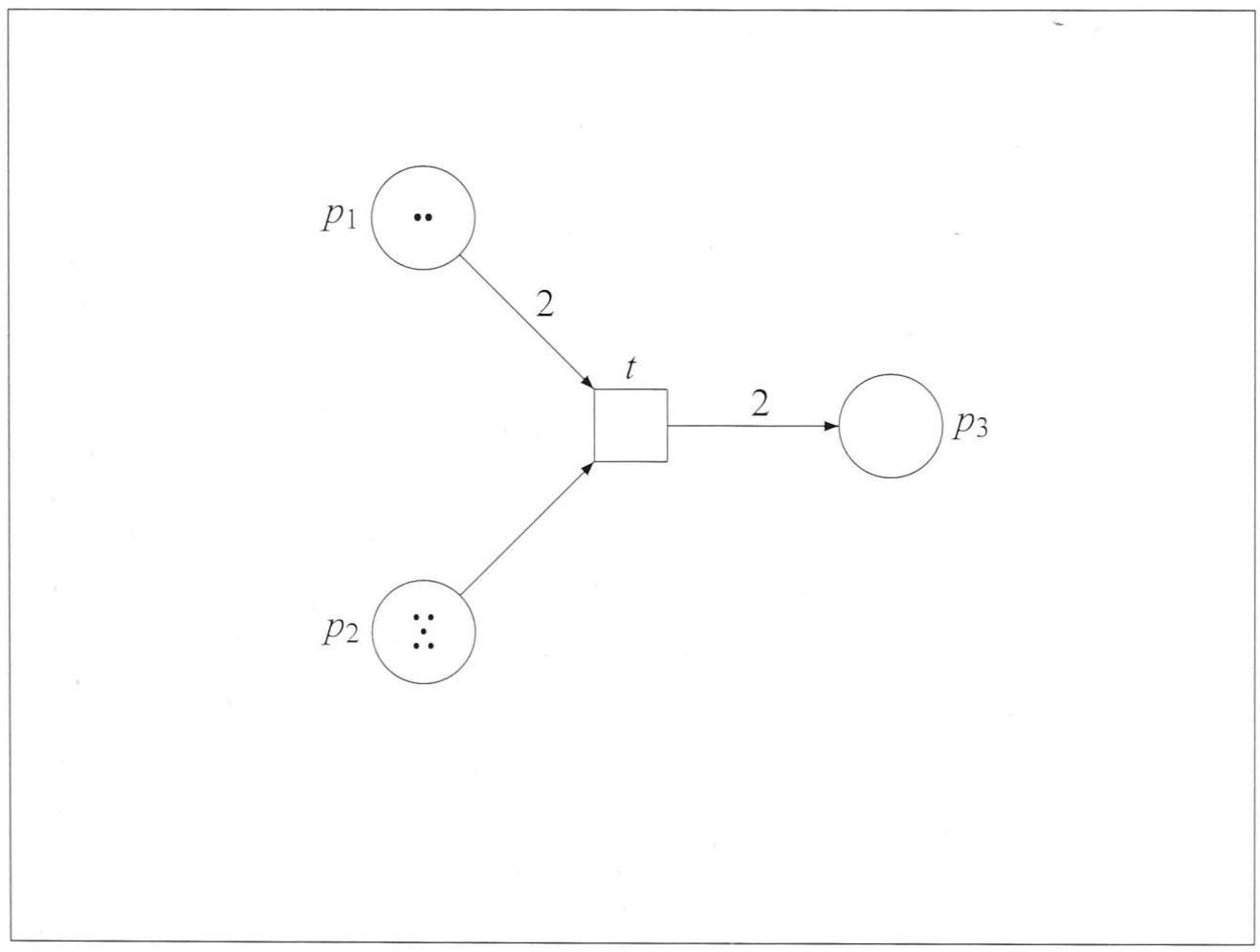


Figure 2.6: Petri net example

In the given example in figure 2.6, taken from [76], we have:

- $P = \{p_1, p_2, p_3\}$.
- $T = \{t\}$.
- $F = \{\langle p_1, t\rangle, \langle p_2, t\rangle, \langle p_3, t\rangle\}$.
- $W = \{\langle p_1, t\rangle \mapsto 2, \langle p_2, t\rangle \mapsto 1, \langle p_3, t\rangle \mapsto 2\}$.
- $M_0 = \{p1 \mapsto 2, p_2 \mapsto 5, p_3 \mapsto 0\}$.

Transitions on a Petri net can be labelled, *i.e.* symbols from an alphabet Σ can be assigned to some or all transitions, resulting in a *labelled* Petri net. It is possible to generate the graph of all reachable markings on a labelled Petri net, which can be equivalently described by an automaton as defined in section 2.2.5.2. Each firing sequence of a labelled Petri net corresponds to a word σ of the alphabet Σ. Generally, we are interested in the languages obtained by all firing sequences of a Petri net or by all firing sequences that reach a given fnal marking.

More details about Petri nets can be found in [43, 76]. One method of doing diagnosis using Petri nets is described in [38].

2.2.6 Diagnosis on Discrete Event Systems

As mentioned in section 2.2.5.1, a language formalism is an appropriate and commonly used mechanism for describing a Discrete Event System and what is happening on it.

Definition 17 (DES System Model)**.** We consider a system Mod whose state can be described as the assignment of *state variables* over a discrete domain, *i.e.* a discrete event system. The evolution of the state variables is also be discrete. The set of all – including unexpected – possible behaviours of this system is a language denoted $\mathcal{L}_{Mod}$ over the set of events Σ that can possibly occur on the system. The set of events is partitioned into *observable* Σ_o and *unobservable* Σ_u events ($\Sigma = \Sigma_o \cup \Sigma_u, \Sigma_o \cap \Sigma_u = \emptyset$).

Definition 18 (DES Observation)**.** The occurrence of an observable event $e_o \in \Sigma_o$ generates an observation o which can be captured or measured on a sensor, or derived from sensor measurements. While the system is running, it generates a flow of observations. We represent the sequence of observations Obs by a language denoted $\mathcal{L}_{Obs}$, where each word in $\mathcal{L}_{Obs}$ is a sequence of observable events consistent with the observations received ($\mathcal{L}_{Obs} \subseteq \Sigma_o^*$). We want to use the sequence of observations Obs to determine the actual sequence of observable events that occured.

The sequence of observable events that occur on the system is a word on Σ_o. However, it is not always possible to determine precisely the sequence of observable events from the sequence of observations received due to a number of factors such as noise, broken sensors or system latency. Hence diagnosis techniques are required to determine what events, both observable and unobservable, occured. Diagnosis is generally considered in the context of failure diagnosis. Faults from

a set of faults Σ_F are possible on the system where $\Sigma_F \subseteq \Sigma$ and particularly $\Sigma_F \subseteq \Sigma_u$ since an observable fault can be trivially diagnosed.

2.2.6.1 Diagnoser and Explanatory Language

To perform diagnosis, an entity called a *diagnoser* is built. A diagnoser can be thought of as an agent in charge of monitoring the observations generated by a system to provide diagnostic reports. A diagnoser is a finite state machine (FSM) built from the system model. It performs diagnosis by estimating the state of the system from a given set of observations. A diagnoser is defined over the set of observable events Σ_o. Hence the language defining a diagnoser is the set of behaviours accepted by the model and consistent with the observations; which we call the *explanatory language*. Formally, we define the explanatory language as:

$$Expl = \mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs} \tag{2.9}$$

where $\otimes$ is the synchronous product as given in defintion 13.

The explanatory language can represented by several tools such as automata or Petri nets. Regardless of the chosen representation, diagnosis still faces the problem of search space explosion.

2.2.6.2 Comparison of Automaton and Petri net approaches

- For diagnosis, both the automaton and Petri net approaches pose structural conditions on the unobservable events. For systems modelled as automata, no cycle of unobservable events may take place after the occurrence of a fault event. The restriction is stronger for Petri net models where the unobservable subnet must be acyclic.

- Automata describe behaviour by explicitly enumerating all possible states and the transitions between them. While this might not be the most elegant, it makes merging of automata by synchronous product easier. Petri nets provide more structure in the representation of transitions. States are not enumerated but state information is distributed among a set of places that capture key conditions that govern the operation of the system.

- Petri nets can be more efficient for highly asynchronous distributed systems (distributed systems are introduced in section 2.2.10).

- It might not be possible to check the *diagnosability* (defined next) of a system using Petri nets when the state space is infinite.

A more detailed comparison of automata and Petri nets is given in [66]. In this thesis, we model DES using automata because they offer a more intuitive way to reason on the systems and are more commonly used in the field.

2.2.7 Diagnosability

While a system is operating, it is important to ensure that it is possible to identify known faults that might occur on the system. How do we know that a diagnosis can be obtained on a system? This can be done by ensuring that the system is designed so that it has a property that guarantees it can be diagnosed (*i.e.* faults can be detected) when it is in operation. We call this property the *diagnosability* of the system.

We give a general definition of diagnosability from [75].

Definition 19 (General Diagnosability)**.** Diagnosability is the property of a partially observable system with a given set of possible faults, that guarantees that those faults can be identified with certainty after a finite set of observations.

Diagnosability is a notion similar to observability or controllability which are more common within the FDI community (*e.g.* see [54]). Diagnosability can be ensured at design time by having the appropriate architecture for the system as well as appropriate number and placement of sensors.

2.2.8 The Reference Approach: Sampath Diagnoser

The seminal work of [90, 91] sets the foundation for the event-based diagnosis of DES. The formalism and semantics are still used within the DX community to date. The system behaviour is modeled as a regular language and is represented using an automaton. The automaton is deterministic and accepts a prefix-closed live language. This means there is at least a transition defined for every state that the system is in and the system cannot reach a point where no event is possible. Observations are assumed to be produced regularly, which means there is no cycle of unobservable events on the system. Diagnosis and diagnosability analysis rely on the construction of a *diagnoser* (as introduced in section 2.2.6.1) which is in the form of an automaton. Each of the states of the diagnoser automaton contains a *label* indicating the current diagnosis.

A set of faults $\Sigma_F = \{f_1, f_2, \ldots, f_m\}$ is considered, where the cardinality of $\Sigma_F = m$. A set of labels is defined as $L = \{N\} \bigcup 2^{\{\Sigma_F \cup \{A\}\}}$. The label N denotes the system is in "normal" operation. When a fault is present, $f_i, i \in \{1, \ldots, m\}$ means that the fault f_i has occurred. The label A means the state of the system is "ambiguous" in the sense that it cannot be determined from the observations whether a fault has occurred or not, or it cannot be distinguished which fault has occurred.

The diagnoser automaton can therefore be defined as $\mathcal{A}_\Delta = \langle Q_\Delta, \Sigma_\Delta, T_\Delta, q_{\Delta 0} \rangle$ where

- $Q_\Delta \subseteq 2^{Q \times L}$ is the finite set of diagnoser states made up of a list of pairs: (system state, label).
- $\Sigma_\Delta = \Sigma_o$, only observable events are considered.
- T_Δ is the set of transitions such that $T_\Delta : Q_\Delta \times \Sigma_\Delta \rightarrow Q_\Delta$.
- $q_{\Delta 0} = \{(q_o, N)\}$ is the initial diagnoser state.

The diagnoser automaton construction is well illustrated in [90]. In this framework, the diagnosability of each fault can be evaluated separately. There might be some faults that are diagnosable and some not. The overall system is diagnosable when all faults are diagnosable.

The diagnoser can be very complex to build for large systems although it can be built offline to resolve the issue. However, it can become impossible to build the diagnoser in cases where the system gets too large and state explosion becomes a problem (the diagnoser is doubly exponential in the number of states [88]). This is the major drawback of the approach.

Many solutions have been proposed to address the state explosion problem. One way is the use of symbolic tools to encode the sytem model in a compact way [39, 93]. In the case of [39], converting to a satisfiability (SAT) problem form allows efficient solving by state of the art SAT solvers.

2.2.9 Decentralised Approach

Besides increasing compactness as mentioned above, another way to address the state explosion problem makes use of the structure of the system. The system model is described in a modular fashion; the global model is given indirectly as a set of components. Diagnosis is performed using a decentralised algorithm that relies on a generic synchronisation operation [83]. Diagnosis is obtained on each set of components making up a *subsystem* and later combined to obtain a global diagnostic result.

Definition 20 (Subsystem)**.** A subsystem γ_i of the global system γ with n components is made up of k components ($k \leq n$). γ_i is given by $\gamma_i = \{c_{i_1}, \ldots, c_{i_k}\}, i_j \in \{1, \ldots, n\}$.

Each component of the system is modelled as an automaton. A set of *communication events* is defined on the system, whose members are used to synchronise (using the synchronous product operation as given in definition 24) the automata representing different components. This decentralised formalism is as expressive as the global automaton one described in [90] with the added advantage of a smaller system model represented with communicating automata. The full global system model can be obtained by synchronising all the automata making up the system on the set of communicating events.

A diagnoser is built for each subsystem, resulting in a set of interactive diagnosers on the system. The diagnosers synchronise with one another to provide a global diagnosis on the system. Due to the underlying global system model, all events emitted system wide are ordered. This allows reasoning on global dependencies among faults. However, this makes it more difficult to handle concurrent systems where we don't necessarily know the ordering of events on the system. A way around this issue to to consider a distributed diagnosis approach, presented next.

2.2.10 Distributed Approach

Distributed diagnosis approaches have been taken by [96, 36]. In this case, only observations on the same component or the same subsystem are ordered. A local diagnoser is defined on

each subsystem and the global diagnosis is computed by the exchange of *messages* (which can be viewed as the common variables or events) between the diagnosers. The global diagnosis is never explicitly calculated like for the decentralised approach. The distributed approach is covered extensively in part I of this thesis. We note that sometimes the terms *decentralised* and *distributed* are used interchangeably in the literature.

2.3 Hybrid Systems

Hybrid systems are dynamical systems modeled with both discrete and continuous aspects. This is relevant to many modern systems which combine logic-based decisions and embedded control actions with physical processes. The evolution of such systems can be captured through mathematical models that in some way bring together the dynamics of the continuous part with the dynamics of the discrete part. There is a variety of mathematical models that can be used but they all somehow basically consist of differential or difference equations for the continuous part and discrete-event models (*e.g.* automata) for the discrete part.

We give an overview of some of the prevalent modeling frameworks used for hybrid systems. Generally, they are a combination of techniques from the FDI and DX communities.

Much of the previous work on hybrid systems focuses on detecting faults by measuring deviations of a dynamical system from the ideal or nominal trajectory. In [54], a residual generating transfer function is created with the property of forcing the residual to become non-zero in the presence of a fault. In [77] a similar approach is used. Both works categorise faults through analysis of the system output, requiring that faults be manifest in the residual error between the system output and a nominal trajectory. A brief introduction to residuals is provided in section 2.2.1.2 including references to in-depth coverage.

In [72] the authors approach diagnosis as a model selection problem. An initial set of qualitative candidate diagnoses are conjectured and then refined using parameter estimation and model fitting techniques. Two continuous system approaches are used to estimate the time to failure t_f and parameters θ_f associated with the conjectured failture mode.

1. Expectation Maximisation (EM) is an iterative technique that converges to optimal values for t_f and θ_f simultaneously.

2. General Likelihood Ratio (GLR) is a technique to estimate the time of failure t_f and then uses the observations obtained after the failure to estimate the fault parameters θ_f by a least squares method.

The assumptions that controller actions are responsible for all events and the inability to deal with multiple sequential faults limits the applicability of this work to dynamic networks where these assumptions hold.

Estimation of continuous dynamics in hybrid systems is particularly challenging and a computationally expensive process because it is necessary to keep track of multiple models and the

transitions between them. Traditionally, Bayesian approaches, such as Kalman filters and Particle filters, have been developed for estimation of continuous dynamics [58, 23]. In these methods, available measured data is used together with prior knowledge about the physical phenomena and the measuring devices in order to sequentially produce estimates of desired dynamics variables while minimising statistical errors.

The Kalman filter is a recursive estimator, *i.e.* it estimates the current state using the previous time step estimate and the current measurement. The Kalman filter provides good tracking results but its application is limited to linear system models with additive Gaussian noises. Most real world systems are non-linear which led to the development of the Extended Kalman filter, an adapted version of the Kalman filter, in order to deal with non-linear dynamics [86]. The Extended Kalman filter involves a linearisation of the non-linear system equations and still requires a well-characterised system model.

If the system model is not well-characterised, then applying the Particle filter for state estimation is more suitable. The Particle filter is a sequential Monte Carlo method that estimates the posterior distribution of the state variables [6]. A set of samples (or particles) are generated to approximate the posterior density function that is then used to update the state estimation. The Particle filtering method is more expensive than the Kalman or Extended Kalman methods for moderately dimensioned state-space systems. Another drawback of the Particle filter is the degeneracy phenomenon whereby after a few iterations all but one particle will have negligible weight. Ways around this include making a good choice of importance density and using resampling techniques [6].

A good coverage and comparison of Kalman and Particle filters are provided in the tutorial given in [81].

In the Hybrid System case, the continuous dynamics estimation problem is significantly complicated by the fact that there are continuous state trajectories to track within multiple discrete modes. Simple extension of the Kalman filter leads to algorithms that require tracking of all possible trajectories and are thus exponential in the number of time steps. One approach [47] gets around this problem by deploying banks of Kalman filters where only trajectories with high confidence are traced. Particle filters are better equipped at dealing with the challenge of tracking continuous system evolution within discrete modes as they can handle high-dimensionality well. The authors in [64] present a methodology for diagnosing hybrid systems based on a particle filtering estimation algorithm. Autonomous transitions are estimated based on complex guard conditions. The algorithm can be used in the case of autonomous transitions, non-linear dynamics and non-Gaussian noise. Convergence of the algorithm depends on the number of particles used which in turn depends on the dimensionality of the continuous state space and number of discrete modes.

The discrete part of a Hybrid System is generally modeled using some variant of a Hybrid Automaton framework first introduced by [46] and found in various work in the literature including [77, 72, 47, 64]. Essentially, a Hybrid Automaton framework brings together continuous dynamics

as described by the state space model described in section 2.2.1 and the discrete modes they fit in as described by the automaton framework introduced in section 2.2.5.2. The work presented in this thesis on Hybrid Systems also uses a hybrid automaton model which is described in more details in Chapter 6.

Part I

Distributed Diagnosis of DES

An overview to the various approaches to diagnosis has been given in chapter 2, first addressed separately by two different communities. There is a recent trend of amalgamating techniques from the two communities to harness strengths from both. This part of the thesis presents a DX-based perspective. It consists of two chapters that are relevant to diagnosis on Discrete Event Systems (DES). Chapter 4 presents a distributed method of doing diagnosis that is based on junction trees. This avoids the computation of a global diagnosis and instead uses the structure of the junction tree to decide on how to synchronise local diagnoses to obtain a globally consistent diagnosis. Chapter 5 is an extension to chapter 4. We augment the distributed junction tree-based approach by ignoring some connections on the system. This helps reduce the complexity, and hence the cost, of the diagnostic reasoning required. However accuracy of the diagnosis is also reduced. We get around this problem by performing an off-line analysis to determine which connections can be safely ignored.

The accuracy analysis presented in chapter 5 has been done in collaboration with Dr. Yannick Pencolé from LAAS, Toulouse, France. The accuracy criterion used here builds on Yannick's previous work as presented in [87].

Chapter 3

Part I: Background

3.1 Motivation

Automation is becoming an increasing part of modern technical systems. As those systems increase in complexity, their supervision becomes more and more challenging. Diagnosis is the process of determining what happened on a system, including the ability to detect faults on the system and if possible isolate the fault(s). Diagnosis is thus an important aspect of the supervision of systems to ensure their smooth running or to take appropriate remedial actions. As systems become more and more complex, new methods to handle diagnosis are required. In *model-based* diagnosis (our focus), the reconstruction of what happened on a system starting from a given initial state, takes into account observations on the system and some sort of model of the considered system. The search space involved in the overall history reconstruction of a system as a monolithic block grows exponentially with the size of the system (*i.e.* the number of variables considered) [88]. Hence for large systems, the diagnostic task becomes intractable. Different approaches, as presented in chapter 2, have been investigated to handle this problem.

There is usually a trade-off between space and time when performing diagnosis. In the Sampath diagnoser [90], the global diagnoser is precompiled offline so that online diagnosis can be done fast. However, the space requirement can get so huge that it can become impossible to apply the technique to large systems. Modern systems are distributed by nature, *i.e.* they are made up of a set of interconnected components. The global behaviour of the system is complex, whereas each component has a simple behaviour. Recent approaches such as [83, 96, 36] take advantage of this distributed nature to avoid computational blow up. Despite requiring more time for online diagnosis compared to the centralised approach, these approaches make it possible to handle larger systems. However, the cited decentralised and distributed methods still do not scale up nicely as system size increases because they involve intermediate merging steps among the local diagnoses before a final diagnostic result is obtained. Merging local diagnoses is expensive in terms of both space and time.

In the ensuing sections of this chapter, underlying concepts, on which are based the work in

chapters 4 and 5, are presented.

3.2 Consistency-based and Abductive Diagnosis

Early diagnostic systems were expert systems that used *shallow* knowledge which consisted of data associated to conclusions by a set of association rules. The knowledge that justifies these associations was not taken into account. This extra knowledge is what is called *deep* knowledge [19] and it allows further explanations that are not possible by using solely shallow knowledge. Model-based diagnosis (MBD) was developed to try to exploit deep knowledge, which involves information concerning structure, functions, behaviour and causality in systems of interest. Deep knowledge can be regarded as more objective information on a system, and in comparison, shallow knowledge (cause-effect patterns) is considered to be more subjective and reliant on an expert's view on system behaviour.

Two main reasoning principles are important for reasoning on deep knowledge for MBD: *consistency* and *abduction*.

Definition 21 (Consistent Diagnosis)**.** An 'explanation' for system behaviour (a diagnosis) is *consistent* when observations on the system match the behaviour predicted by the model after synchronisation of the system model and observations obtained on the system.

Definition 21 defines consistency-based diagnosis. It allows us to determine which component(s) is(are) faulty. It is obtained by checking the diagnosis obtained on a system match the observations on the system. Consistency-based diagnosis has been systemised by [85].

On the other hand, abductive diagnosis requires the availability of fault models.

Definition 22 (Abductive Diagnosis)**.** An *abductive* diagnosis explains a symptom by finding a set of causes that logically imply the symptom itself.

For abductive diagnosis to be possible, the system model must contain a description of what happens in the presence of a failure, with possible distinction of different failure types.

The two types of diagnostic reasoning converge in the attempt to exploit information about both normal behaviour and faulty behaviour to provide a fuller diagnosis. Hence both approaches were integrated ([26, 51, 9]) and shown to be the two extremes of a wide spectrum of possible definitions of diagnosis [27].

In our approach, we assume that normal behaviour as well as fault models are available and hence diagnosis is both consistency-based and abductive.

3.3 The Diagnosis Problem and Global Consistency

A definition of the concept of *diagnosis* is given. It will form the basis of what is understood by the term *diagnosis* throughout this thesis. Before giving the definition for *diagnosis*, we define essential related concepts.

As mentioned in Chapter 2, the evolution of discrete event systems can be represented by a language framework. An event can be represented by a symbol on a given alphabet. Languages can be *synchronised* to make them consistent (define in lit review section) with one another.

Definition 23 (Synchronisation)**.** The synchronisation between two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ is an operation that computes the words on $\mathcal{L}_1$ that are consistent with the words in $\mathcal{L}_2$ given a correspondence between the alphabets of the two languages. In other words, synchronisation involves computing the projection of language $\mathcal{L}_1$ over $\mathcal{L}_2$ and vice versa, and is carried out through the synchronous product $\otimes$, an operation introduced in chapter 2 in definition 13.

Definition 24 (Synchronous Product)**.** The synchronous product $\otimes$ between two languages $\mathcal{L}_1$ over Σ_1 and $\mathcal{L}_2$ over Σ_2 computes all the words over $\Sigma_1 \cup \Sigma_2$ whose projection on Σ_i is $\mathcal{L}_i$: $\mathcal{L}_1 \otimes \mathcal{L}_2 = \{\sigma \in (\Sigma_1 \cup \Sigma_2)^\star \mid \forall i \in \{1,2\},\ P_{\Sigma_1 \cup \Sigma_2 \to \Sigma_i}(\sigma) \in \mathcal{L}_i\}$.

Definition 25 (Diagnosis)**.** Diagnosis is the problem of finding out what happened on a system given a model of the system and observations on the system. Hence a diagnosis returns a set of possible hypotheses (diagnosis candidates) that might explain the observations made on the system. Given a set of observation *Obs* on the system, a diagnosis Δ for *Obs* is the set of all diagnosis candidates $\{d_1, d_2, \ldots, d_n\}$ that explain *Obs*.

In the case of model-based diagnosis, we assume that a model *Mod* of the system is available. Therefore we can consider that diagnosis is the *synchronisation* of the model and the observations. If $\mathcal{L}_\Delta$ is the language that describes a diagnosis Δ, $\mathcal{L}_{Mod}$ is the language that describes the system model and $\mathcal{L}_{Obs}$ is the language that describes observations on the system, then we can write

$$\mathcal{L}_\Delta = \mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs} \tag{3.1}$$

We call $\mathcal{L}_\Delta$ the explanatory language because it provides an explanation of what happened on the system *Mod* given the observation *Obs*.

3.3.1 Proof that $\mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs}$ is a solution to the diagnosis problem

Given that $\mathcal{L}_{Obs}$ describes a sequence of observations on a given system *Mod*, then the synchronisation of $\mathcal{L}_{Mod}$ with $\mathcal{L}_{Obs}$ finds all the words on $\mathcal{L}_{Mod}$ that are consistent with those in $\mathcal{L}_{Obs}$. Since $\mathcal{L}_{Mod}$ describes everything that can possibly happen on the system, and assuming that *Mod* is complete, then $\mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs}$ returns all the scenarios that could possibly have occurred on the system given the observations *Obs*. Hence $\mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs}$ is a solution to the diagnosis problem.
$\mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs} = \{\sigma \in (\Sigma_{Mod} \cup \Sigma_{|Obs})^\star \mid \forall i \in \{Mod, Obs\},\ P_{\Sigma_{Mod} \cup \Sigma_{Obs} \to \Sigma_i}(\sigma) \in \mathcal{L}_i\}$.

3.3.1.1 Proof that a globally consistent distributed diagnosis is a solution to the diagnosis problem

We are interested in showing that a *globally consistent* diagnosis over a distributed system is a solution to the diagnosis problem as defined in definition 25. We first define the concept of *global consistency* with respect to languages. Since a diagnosis over a system is associated with an explanatory language, the concept can then be extended to the diagnosis context.

Definition 26 (Global Consistency)**.** Given a set $\mathcal{S}$ of languages $\{\mathcal{L}_1, \ldots, \mathcal{L}_m\}$ that describe the behaviour of m clusters formed with n components. The language describing $\mathcal{S}$, and defined over $\Sigma_1 \cup \cdots \cup \Sigma_m$, can be written as $\mathcal{L}_\mathcal{S} = \mathcal{L}_1 \otimes \cdots \otimes \mathcal{L}_m$. We say that the set $\mathcal{S}$ and its elements are *globally consistent* if $\forall i, \mathcal{L}_i = \mathrm{P}_{\Sigma_\mathcal{S} \to \Sigma_i}(\mathcal{L}_\mathcal{S})$.

A distributed system $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ composed of components $\gamma_1, \ldots, \gamma_n$ can be implicitly defined as a set of languages $\mathcal{L}_{Mod} = \mathcal{L}_{Mod1} \otimes \cdots \otimes \mathcal{L}_{Modn}$. $\mathcal{L}_{Mod}$ does not need to be explicitly calculated since the behaviour of the system is already captured within the models of the constituent components. Similarly, the observations on the system can be defined by $\mathcal{L}_{Obs} = \mathcal{L}_{Obs1} \otimes \cdots \otimes \mathcal{L}_{Obsn}$.

From definition 3.1, for a distributed system Γ, we can write:

$$\begin{aligned}
\mathcal{L}_\Delta &= \mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs} && (3.2)\\
&= (\mathcal{L}_{Mod1} \otimes \cdots \otimes \mathcal{L}_{Mod1}) \otimes (\mathcal{L}_{Modn} \otimes \cdots \otimes \mathcal{L}_{Modn}) && (3.3)\\
&= (\mathcal{L}_{Mod1} \otimes \mathcal{L}_{Obs1}) \otimes \cdots \otimes (\mathcal{L}_{Modn} \otimes \mathcal{L}_{Obsn}) && (3.4)\\
&= \mathcal{L}_{\Delta 1} \otimes \cdots \otimes \mathcal{L}_{\Delta n} && (3.5)
\end{aligned}$$

From equation 3.5, we see that the explanatory language $\mathcal{L}_\Delta$ of the distributed system Γ can be obtained by synchronising the explanatory languages $\mathcal{L}_{\Delta 1}, \ldots, \mathcal{L}_{\Delta n}$ of its components. The explanatory language $\mathcal{L}_\Delta$ can thus be considered as a set of languages; $\mathcal{L}_\Delta : \{\mathcal{L}_{\Delta 1}, \ldots, \mathcal{L}_{\Delta n}\}$. If $\mathcal{L}_\Delta$ is *globally consistent*, then $\forall i, i \in \{1, \ldots, n\}, \mathcal{L}_{\Delta i} = \mathrm{P}_{\Sigma_\Gamma \to \Sigma_i}(\mathcal{L}_\Delta)$.

We illustrate the global consistency concept using an example consisting of 3 languages, each represented by an automaton (see figure 3.1). Each language represent a diagnosis of its representative component. Events o_i are observable events. Events f, u_i and e_i are different classes of unobservable events: f is a fault event, e_i events can be shared among components, u_i events are internal events that cannot be shared.

Let the automata A, B and C represent respectively the explanatory languages $\mathcal{L}_{\Delta A}$, $\mathcal{L}_{\Delta B}$ and $\mathcal{L}_{\Delta C}$ of three components A, B and C of a given system. The diagnostic hypotheses for each component and their synchronisation are given in table 3.1.

The three automata A, B and C are not globally consistent. When we look at automaton A, we have 2 diagnostic hypotheses: $\{ f.e_1 , u_1.e_2 \}$. The fault event f appears in one hypothesis but not

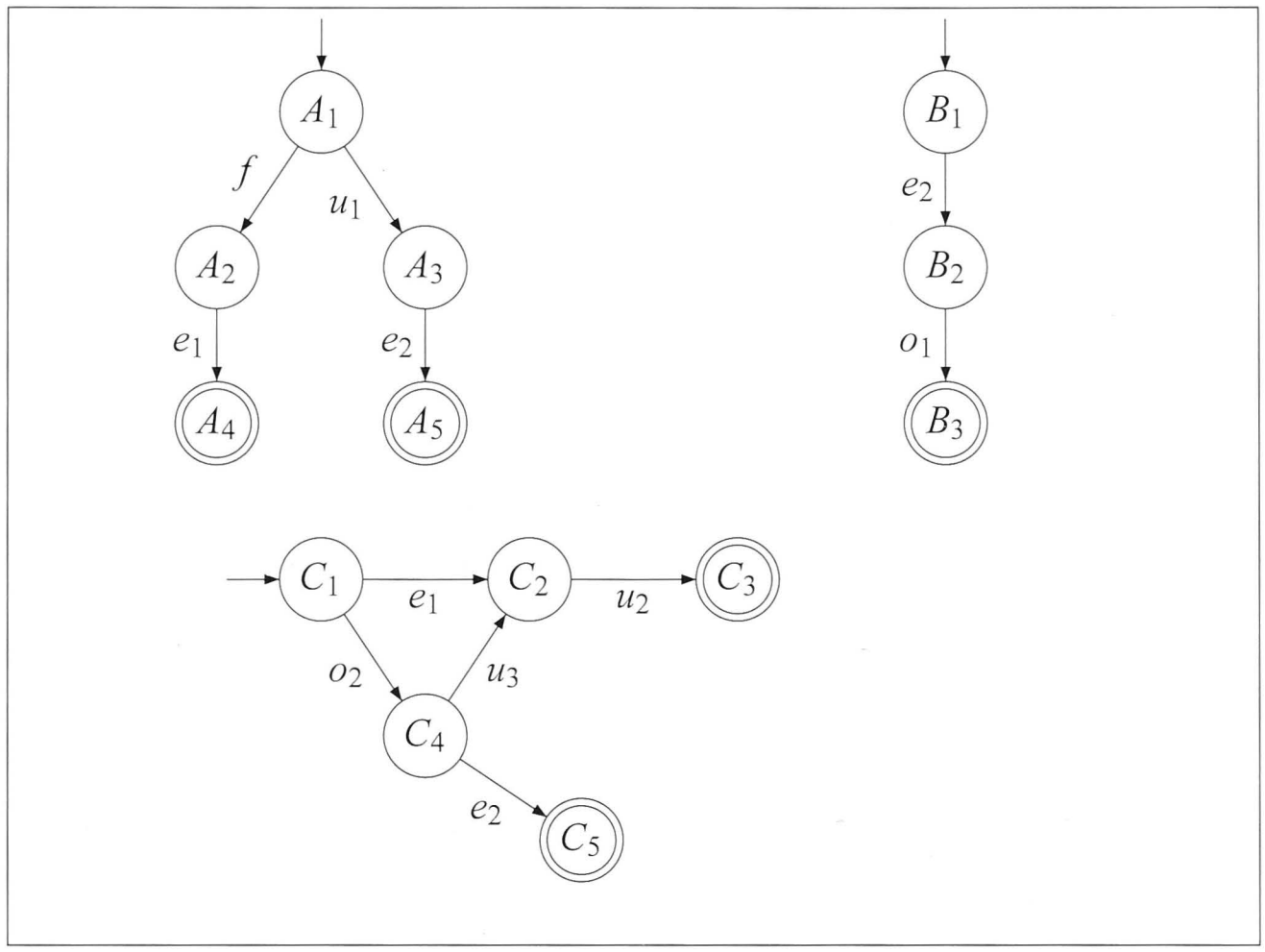


Figure 3.1: Example of distributed system consisting of three components A, B and C modeled by automata

Component	Diagnostic Hypotheses
A	$\{ f.e_1 , u_1.e_2 \}$
B	$\{ e_2.o_1 \}$
C	$\{ e_1.u_2 , o_2.u_3.u_2, o_2.e_2 \}$
A,B,C	$\{ u_1.o_2.e_2.o_1 , o_2.u_1.e_2.o_1 \}$

Table 3.1: Diagnostic hypotheses for components A, B and C and for their synchronised product

in the other and hence we cannot be sure whether it happened or not. Similarly, when looking at automaton C alone, there are three hypotheses and we do not know which happened. However, when we synchronise all the automata, we are able to make $\mathcal{L}_{\Delta A}$, $\mathcal{L}_{\Delta B}$ and $\mathcal{L}_{\Delta C}$ consistent with one another, *i.e.* to achieve global consistency. We obtain the globally consistent explanatory language $\mathcal{L}_{\Delta}{}' = \{ u_1.o_2.e_2.o_1 , o_2.u_1.e_2.o_1 \}$. The ordering of the occurrence of u_1 and o_2 is not very important as it does not affect the projection back onto the component languages. Thus, we can project back to obtain the globally consistent languages $\mathcal{L}_{\Delta A}$, $\mathcal{L}_{\Delta B}$ and $\mathcal{L}_{\Delta C}$:

$$\begin{aligned} \mathcal{L}_{\Delta'_A} &= \{u_1.e_2\} \\ \mathcal{L}_{\Delta'_B} &= \{e_2.o_1\} \\ \mathcal{L}_{\Delta'_C} &= \{o_2.e_2\} \end{aligned}$$

This result tells us that f did not occur on the system. Hence, a globally consistent diagnosis on a distributed system is a solution to the diagnosis problem. In this thesis, we strive to obtain a globally consistent diagnosis in order to compute the diagnosis of a distributed system. Algorithms and techniques are presented in the next two chapters that allow us to avoid the computation of a global diagnosis by using the global consistency property.

Chapter 4

Diagnosis with Junction Trees

We present in this chapter a method for doing distributed diagnosis on Discrete Event Systems that minimises the number of local merges required to obtain a global diagnosis. The method utilises the knowledge (for which a proof was presented in Chapter 3) that a globally consistent diagnosis over a distributed system is a solution to the diagnosis problem. The complexity of numerous algorithms in different domains drops when applied to trees. In the case of the global consistency algorithm, ensuring local consistency on a tree also ensures global consistency while also being less expensive to compute than a global diagnosis. A popular solution to convert a graph into a tree is to make it into a *junction tree* [49], where the vertices are gathered in clusters. We thus transform the topological graph of the system into a junction tree where each cluster corresponds to a subsystem. Clusters can be synchronised locally to obtain a globally consistent diagnosis. The proposed diagnosis method is applied to different classes of networks to investigate performance on them.

4.1 Preliminaries

We are interested in Discrete Event Systems and take a DX Event-based approach, as introduced in chapter 2. We use the language formalism framework described in section 2.2.5. For ease of reading, we reproduce here the definition for *Projection* on languages (given in definition 12) and the definition for *Synchronous Product* (given in definition 13).

Definition12 (Projection) The projection on Σ' of a word σ over $\Sigma \supseteq \Sigma'$ denoted $P_{\Sigma\to\Sigma'}(\sigma)$ keeps all the elements of σ in Σ'. Formally,

$$P_{\Sigma\to\Sigma'}(\sigma) = \begin{cases} \varepsilon & \text{if } \sigma = \varepsilon \\ P_{\Sigma\to\Sigma'}(\sigma') & \text{if } \sigma = e.\sigma' \text{ and } e \in \Sigma \setminus \Sigma' \\ e.P_{\Sigma\to\Sigma'}(\sigma') & \text{if } \sigma = e.\sigma' \text{ and } e \in \Sigma' \end{cases}$$

The projection on Σ' of a language $\mathcal{L}$ over Σ is denoted $P_{\Sigma\to\Sigma'}(\mathcal{L})$ and defined by $\{P_{\Sigma\to\Sigma'}(\sigma) \mid \sigma \in \mathcal{L}\}$. The inverse operation $P^{-1}_{\Sigma\to\Sigma'}$ of the projection from Σ to Σ' generates all the finite words on Σ

whose projection on Σ' is in the parameter: $P^{-1}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\sigma \in \Sigma^\star \mid P_{\Sigma\to\Sigma'}(\sigma) \in \mathcal{L}\}$.

Projection is used to describe important operations on languages as follows.

Definition 13 (Synchronous Product) The synchronous product $\otimes$ between two languages $\mathcal{L}_1$ over Σ_1 and $\mathcal{L}_2$ over Σ_2 computes all the words over $\Sigma_1 \cup \Sigma_2$ whose projection on Σ_i is $\mathcal{L}_i$: $\mathcal{L}_1 \otimes \mathcal{L}_2 = \{\sigma \in (\Sigma_1 \cup \Sigma_2)^\star \mid \forall i \in \{1,2\},\ P_{\Sigma_1\cup\Sigma_2\to\Sigma_i}(\sigma) \in \mathcal{L}_i\}$.

We now introduce a formal definition for the *local consistency* operation using the language formalism. *Local consistency* as a concept is described in more details in section 4.2.4.

Definition 27 (Local Consistency)**.** The local consistency operation of language $\mathcal{L}_1$ over Σ_1 on $\mathcal{L}_2$ over Σ_2 denoted $cons_{\Sigma_1,\Sigma_2}(\mathcal{L}_1, \mathcal{L}_2)$ returns the minimum sublanguage of $\mathcal{L}_2$ such that the synchronous product with $\mathcal{L}_1$ is not modified: $cons_{\Sigma_1,\Sigma_2}(\mathcal{L}_1, \mathcal{L}_2) = \{\sigma \in \mathcal{L}_2 \mid P_{\Sigma_2\to\Sigma_1\cap\Sigma_2}(\sigma) \in P_{\Sigma_1\to\Sigma_1\cap\Sigma_2}(\mathcal{L}_1)\}$ or equivalently $cons_{\Sigma_1,\Sigma_2}(\mathcal{L}_1, \mathcal{L}_2) = \mathcal{L}_2 \cap P^{-1}_{\Sigma_2\to\Sigma_1\cap\Sigma_2}(P_{\Sigma_1\to\Sigma_1\cap\Sigma_2}(\mathcal{L}_1))$.

4.2 Model-based Diagnosis of Discrete-Event Systems

We use the same DES framework described in section 2.2.6, which is a known and well-used framework used in the DX community. We assume we have a system model Mod as defined in definition 17, and that the system generate a sequence of observations Obs as defined in definition 18. We are interested in obtaining a *diagnosis* Δ of the system as defined in definition 25. We are interested in the explanatory language $\mathcal{L}_\Delta$ which tells us what happened on the system Mod given the sequence of observations Obs. The explanatory language is given in equation 3.1 and reproduced here:

$$\mathcal{L}_\Delta = \mathcal{L}_{Mod} \otimes \mathcal{L}_{Obs}$$

Without loss of meaning, we write a simplified version of equation 3.1 for ease of notation going forward:

$$\Delta = \text{Mod} \otimes \text{Obs} \tag{4.1}$$

Languages can be represented by several tools. Regular languages are often represented by automata or Petri nets. The problem with these tools is that of *state explosion*. The size of these structures is exponential in the number of state variables, which makes them difficult to use in practice.

We use automata for our DES models. Automata describe behaviour by explicitly enumerating all possible states and the transitions between them. While this might not be the most elegant, it makes merging of automata by synchronous product easier.

Petri nets provide more structure in the representation of transitions. States are not enumerated but state information is 'distributed' among a set of *places* that capture key conditions that govern the operation of the system.

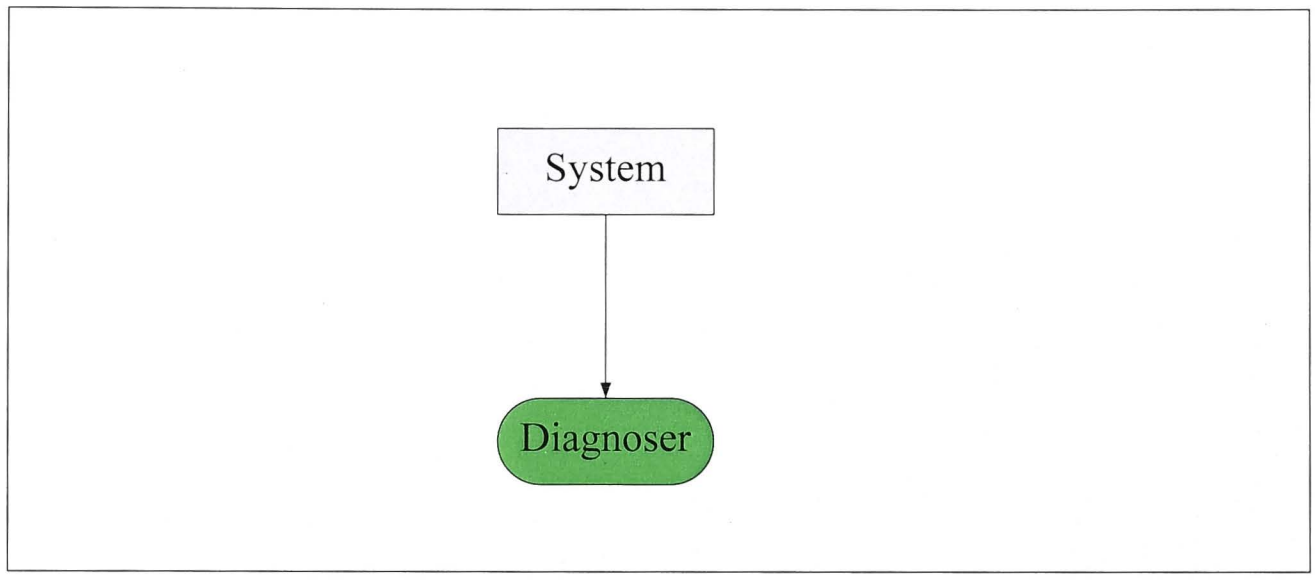


Figure 4.1: Global Diagnoser Approach

4.2.1 Distributed Approach

Real-world systems are often distributed by nature, *i.e.* they consist of a set of interconnected components. The global behaviour of the system is complex, whereas each component has a simple behaviour. Recent approaches take advantage of this distributed nature to avoid computational blow up.

4.2.2 Distributed Modeling

Modern technical systems are usually formed by combining simple components with simple behaviours leading to a device that exhibits complex behaviours. Rather than modeling the whole system, it is often preferable to model each component separately for many good reasons: fewer chances to make mistakes or forget behaviours, reusability, compactness.

Since the system is a set of components, each component γ_i can be modeled separately: Mod_i defined on alphabet Σ_i. Some formalisms assume that components share variables. Here, without loss of generality, we assume that components share events such that an event shared by several components must occur on each component at the same time. Other events may occur in a completely concurrent manner.

The system $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ composed of components $\gamma_1, \ldots, \gamma_n$ is modeled as a set of languages $d\text{Mod} = \{\text{Mod}_1, \ldots, \text{Mod}_n\}$ over the alphabets $\Sigma_1, \ldots, \Sigma_n$. The global model of the system is implicitly defined by $\text{Mod} = \text{Mod}_1 \otimes \cdots \otimes \text{Mod}_n$ but never explicitly computed.

4.2.3 Distributed Diagnosis and Global Consistency

Global consistency of a diagnosis means that there is no conflict in the diagnostic conclusion reached no matter which part or component of the system being diagnosed is being looked at. The three approaches taken to consistency-based diagnosis have been:

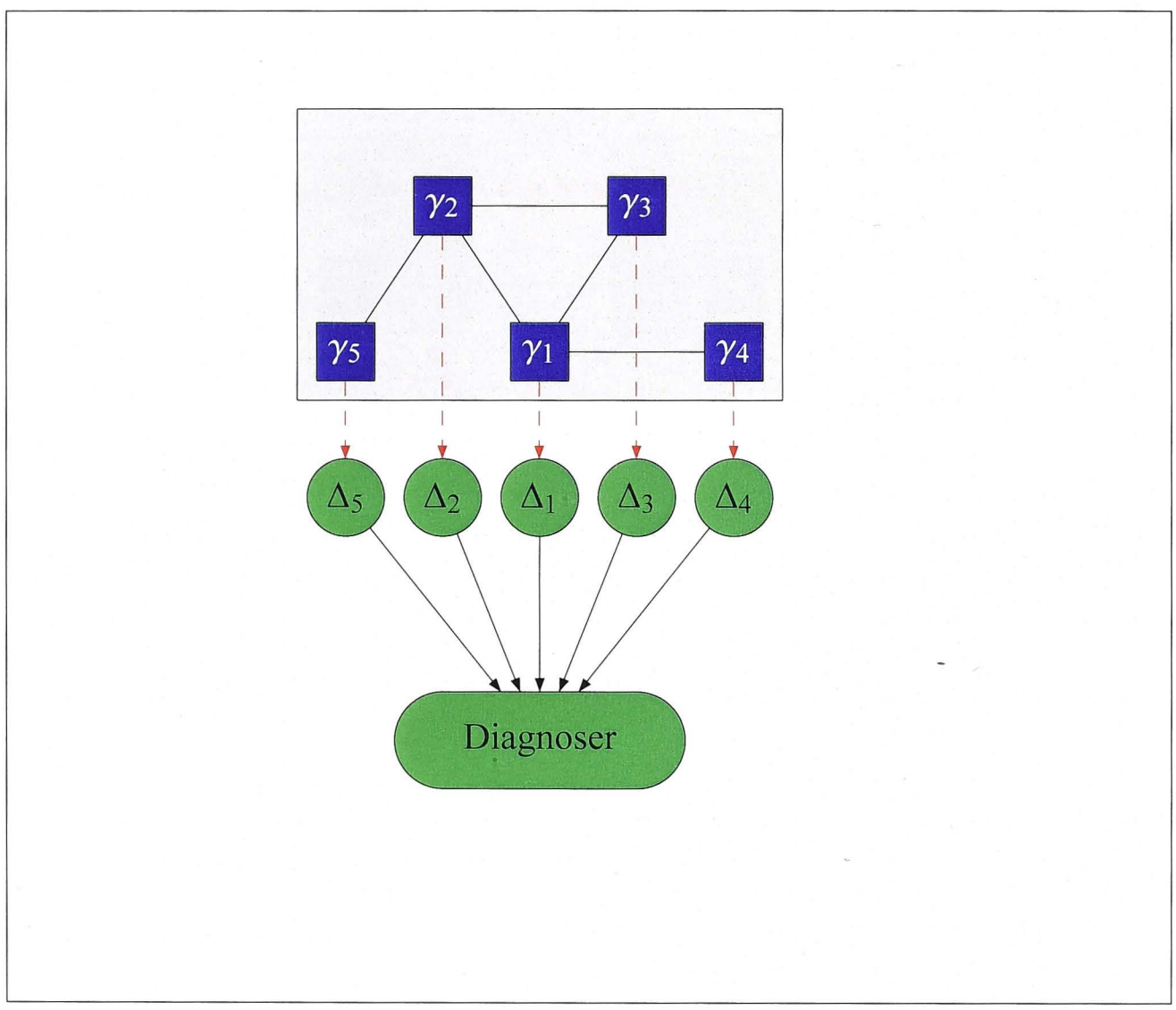


Figure 4.2: Decentralised Diagnoser Approach

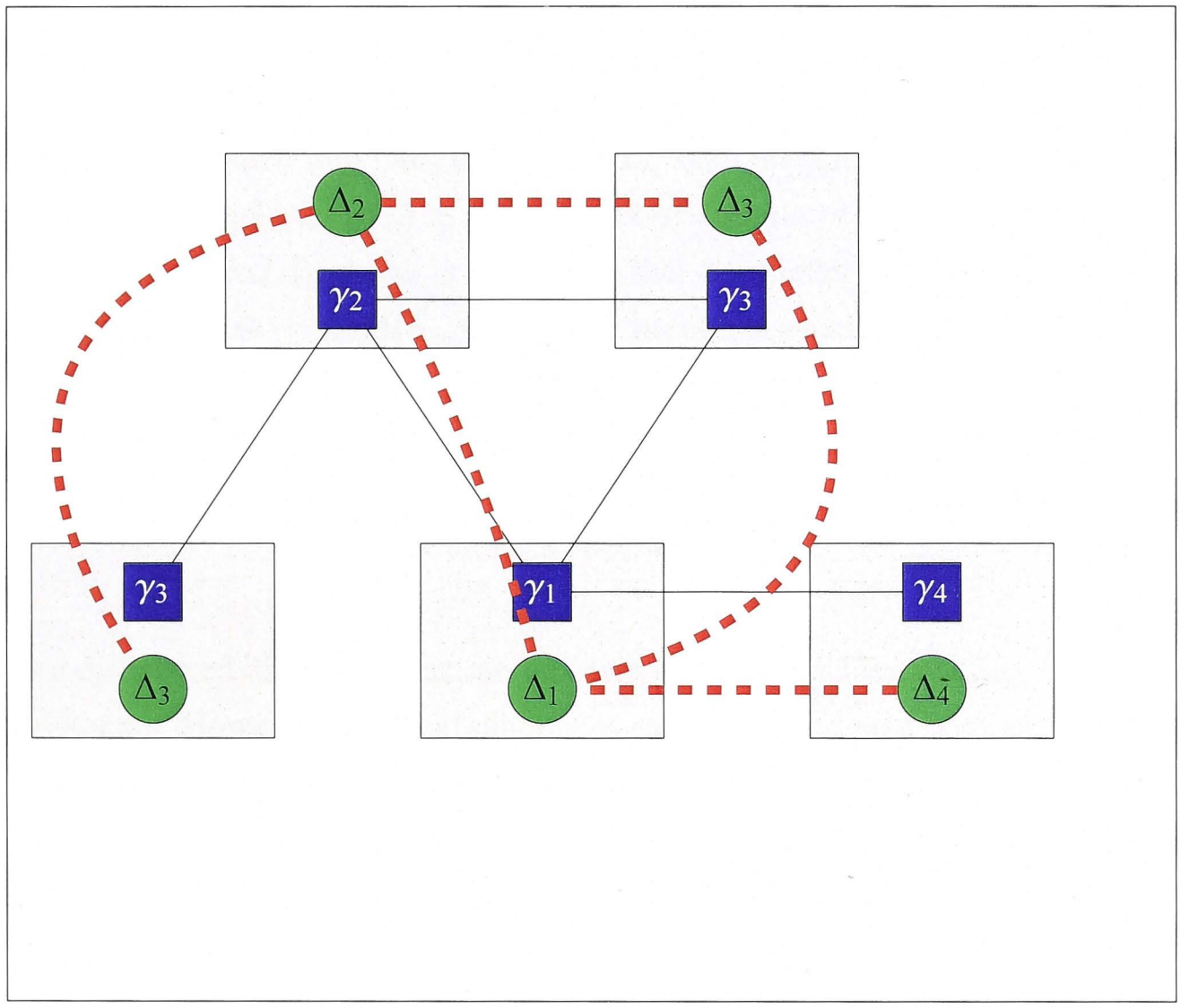


Figure 4.3: Distributed Diagnoser Approach (the dotted red line denotes local consistency between diagnoses)

1. to calculate a global diagnosis using a global diagnoser defined over the whole system [90] (Figure 4.1);
2. to calculate a global diagnosis using a decentralised diagnoser that gathers and merges information from a set of local diagnosers [83] (Figure 4.2);
3. to *avoid* calculating a global diagnosis but to instead ensure consistency among local diagnosers [96, 59] (Figure 4.3).

4.2.3.1 Global Consistency defined on a distributed model

The alphabet Σ_i that represents the events of each component γ_i is partitioned into observable events Σ_{io} and unobservable events Σ_{iu}. Moreover, we assume that the global observations Obs on the system can be distributed into Obs_i defined on Σ_{io} such that $\text{Obs} = \text{Obs}_1 \otimes \cdots \otimes \text{Obs}_n$.

A *distribution* $\mathcal{S} = \{S_1, \ldots, S_m\} \in 2^{2^\Gamma}$ is a set of subsets of Γ such that $\mathcal{S}$ covers Γ: $S_1 \cup \cdots \cup S_m = \Gamma$. A *distributed diagnosis* is a mapping that associates with each subset S_i a diagnosis $d\Delta(S_i)$ such that $d\Delta(S_1) \otimes \cdots \otimes d\Delta(S_m) = \Delta$. The literature usually assumes that $\mathcal{S}$ is a partition of Γ [83].

The local diagnoses can be simply computed by:

$$d\Delta(S_i) = \bigotimes_{\gamma_k \in S_i} (\text{Mod}_k \otimes \text{Obs}_k). \tag{4.2}$$

This returns a distributed diagnosis that can be easily computed as long as any S_i contains a small number of elements. However, the local diagnoses can be inconsistent with each other. Basically, some words of $d\Delta(S_i)$ should be removed because they disappear when S_i is synchronised with other S_j elements. Thus, we are interested by the globally consistent distributed diagnosis:

A distributed diagnosis $d\Delta$ is *globally consistent* if $\forall i \in \{1, \ldots, m\}$, $d\Delta(S_i) = P_{\Sigma \to \Sigma_{S_i}}(\Delta)$ where $\Sigma_{S_i} = \bigcup_{\gamma_k \in S_i} \Sigma_k$.

The globally consistent distributed diagnosis is such that no word of any $d\Delta(S_i)$ can be removed. We want to compute this *refined* distributed diagnosis but the goal is to avoid the computation of Δ.

4.2.4 Local Consistency

The local consistency property requires that any pair of local diagnoses are consistent. Formally, a distributed diagnosis $d\Delta$ is locally consistent if $\forall \{S_1, S_2\} \subseteq \mathcal{S}$, $P_{\Sigma_{S_1} \to \Sigma_{S_1} \cap \Sigma_{S_2}}(d\Delta(S_1)) = P_{\Sigma_{S_2} \to \Sigma_{S_1} \cap \Sigma_{S_2}}(d\Delta(S_2))$.

It is possible to refine a distributed diagnosis using local consistency as presented in Algorithm 1. After the distribution is performed, and a local diagnosis is computed for each subsystem, the algorithm takes pairs of subsystems and performs a local consistency on these diagnoses. Basically, the idea is to remove the word of $d\Delta(S_1)$ that cannot be synchronised with any word of

$d\Delta(S_2)$, and vice versa. The local consistencies can actually be performed in any order.

Provided that the distribution structure of the system can be represented by a tree, the algorithm presented in Algorithm 1 terminates and is sound.

Algorithm 1 Distributed diagnosis algorithm based on local consistency

1: **input** $\Gamma, \{\text{Mod}_1, \ldots, \text{Mod}_n\}, \{\text{Obs}_1, \ldots, \text{Obs}_n\}$
2: $\mathcal{S} = \{S_1, \ldots, S_m\} := \textit{distribution}(\Gamma)$
3: **for all** $i \in \{1, \ldots, m\}$ **do**
4: $\quad d\Delta(S_i) = \bigotimes_{\gamma_k \in S_i} (\text{Mod}_k \otimes \text{Obs}_k)$
5: **repeat**
6: $\quad$ **for all** $\{S_1, S_2\} \subseteq \mathcal{S}$ **do**
7: $\quad\quad d\Delta(S_2) := cons_{\Sigma_{S_1}, \Sigma_{S_2}}(d\Delta(S_1), d\Delta(S_2))$
8: $\quad\quad d\Delta(S_1) := cons_{\Sigma_{S_2}, \Sigma_{S_1}}(d\Delta(S_2), d\Delta(S_1))$
9: **until** $d\Delta$ is stable

However, as shown in [96], local consistency does not ensure global consistency. Moreover, because the languages may be infinite, no fix-point is reached in the worst case; the algorithm does not terminate. As noticed by the authors in [96] , both problems disappear when the topology of the system forms a tree.

Definition 28 (Graph)**.** A graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ consists of a set of vertices $\mathcal{V}$ (also called *nodes*) linked together by a set of edges $\mathcal{E}$. Each edge $edg \in \mathcal{E}$ is connected to a pair of vertices (ver_i, ver_j) one on each side.

In a *simple graph*, there are no self-loops, *i.e.* $i \neq j$ and no repeated edge, *i.e.* $\forall p, \forall q, if\ \exists edg_p = (ver_i, ver_j),\ then\ if\ edg_q = (ver_i, ver_j), q = p$. A graph can be *directed* or *undirected*. In a *directed* graph, the direction in which two nodes are connected matters. Detailed explanations can be found in textbooks on graph theory such as [10].

A *topology* of a distributed representation $\mathcal{S}$ of the system is a graph (as defined in Definition 28) $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V} = \mathcal{S}$ is the set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a symmetric and anti-reflexive set of edges such that $\forall \{S, S'\} \subseteq \mathcal{V}$, $\forall e \in \Sigma_S \cap \Sigma_{S'}$, $\exists S_0, \ldots, S_{k+1}$ such that:

- $S_0 = S$ and $S_{k+1} = S'$
- $\forall i \in \{1, \ldots, k\},\ e \in \Sigma_{S_i}$, and
- $\forall i \in \{0, \ldots, k\},\ \langle S_i, S_{i+1} \rangle \in \mathcal{E}$

Two subsystems that share an event are connected through an edge, or through a chain of edges where intermediate subsystems also share this event.

Definition 29 (Tree)**.** The graph $\mathcal{G}$ is a tree if for any pair S_i and S_j, there is exactly one path on the graph that contains no loop and leads from S_i to S_j.

Provided that the distribution of the system can be represented by a tree, the algorithm presented above terminates and is sound.

Theorem 4.2.1. *Given that the distribution of a system generates a tree $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. If the local diagnosis $d\Delta(S_i)$ is computed for each subsystem $S_i \in \mathcal{V}$ and the local consistency procedure is applied until stability is reached, then for a distributed system that is locally consistent, it is also globally consistent. i.e. $P_{\Sigma \to \Sigma_i}(\Delta) = d\Delta(S_i)$.*

We give a proof of this result. Similar proofs can be found in [96] with slightly different definition of the topology. In particular in [96], an edge connects two vertices whenever these two vertices share an event, while here we only required them to be connected through a chain of vertices that share this event.

Proof. Let $X \subseteq \mathcal{V}$ be a subset of subsystems, we denote $\Sigma_X = \bigcup_{S \in X} \Sigma_S$ and $\mathcal{L}_X = \bigotimes_{S \in X} d\Delta(S)$.

1. We start with $X = \{S_i\}$. We add $S_k \notin X$ incrementally such that $S_j \in X$ and $\langle S_k, S_j \rangle \in \mathcal{E}$. We prove by induction that proposition α that states that for any $S_p \in X$, $P_{\Sigma_X \to \Sigma_p}(\mathcal{L}_X) = d\Delta(S_p)$ is true.

2. This is clearly the case for $X = \{S_i\}$.

3. Let $X' = X \cup \{S_k\}$, $S_j \in X$ and $\langle S_k, S_j \rangle \in \mathcal{E}$. We first show that $\mathcal{L}_X$ and $d\Delta(S_k)$ are locally consistent.
   We note that because of the definition of $\mathcal{G}$ and since $\mathcal{G}$ is a tree, $\Sigma_X \cap \Sigma_k = \Sigma_j \cap \Sigma_k$. We can thus write:
   $P_{\Sigma_X \to \Sigma_X \cap \Sigma_k}(\mathcal{L}_X)$

   $= P_{\Sigma_X \to \Sigma_j \cap \Sigma_k}(\mathcal{L}_X)$ (because $\Sigma_X \cap \Sigma_k = \Sigma_j \cap \Sigma_k$)

   $= P_{\Sigma_j \to \Sigma_j \cap \Sigma_k}(P_{\Sigma_X \to \Sigma_j}(\mathcal{L}_X))$ (since $\Sigma_j \cap \Sigma_k \subseteq \Sigma_j \subseteq \Sigma_X$)

   $= P_{\Sigma_j \to \Sigma_j \cap \Sigma_k}(d\Delta(S_j))$ (by induction)

   $= P_{\Sigma_k \to \Sigma_j \cap \Sigma_k}(d\Delta(S_k))$ (by local consistency)

   $= P_{\Sigma_k \to \Sigma_X \cap \Sigma_k}(d\Delta(S_k))$ (because $\Sigma_X \cap \Sigma_k = \Sigma_j \cap \Sigma_k$)

   Thus, $\mathcal{L}_X$ and $d\Delta(S_k)$ are locally consistent. This also means that for any $S_p \in X$, $P_{\Sigma_{X'} \to \Sigma_p}(\mathcal{L}_{X'}) = P_{\Sigma_X \to \Sigma_p}(\mathcal{L}_X) = d\Delta(S_p)$, and $P_{\Sigma_{X'} \to \Sigma_k}(\mathcal{L}_{X'}) = d\Delta(S_k)$.

Hence, for $X = \mathcal{V}$, we have the following result: $\forall S_i \in \mathcal{S}$, $P_{\Sigma \to \Sigma_i}(\Delta) = d\Delta(S_i)$. The distributed diagnosis is then globally consistent. □

We propose to build such a distribution of the system, using the junction tree theory.

4.3 Diagnosis by Junction Tree

4.3.1 Junction Tree

The concept of the junction tree is borrowed from the field of probabilistic inference where its structure is useful for working in complex domains [49]. Note that *junction trees* are also referred to as *join trees* in the literature [92].

Definition 30 (Junction Tree). Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a graph. A *junction tree* for $\mathcal{G}$ is a pair $(\mathcal{T}, C)$, where $\mathcal{T}$ is a tree and C is a function which maps each node i in tree $\mathcal{T}$ into a label C_i called a *cluster*. The junction tree must satisfy the following properties:

1. $C_i \subseteq \mathcal{V}$, *i.e.* each cluster is a set of vertices from $\mathcal{G}$.

2. If two vertices are connected in $\mathcal{G}$, they will appear together in some cluster C_i.

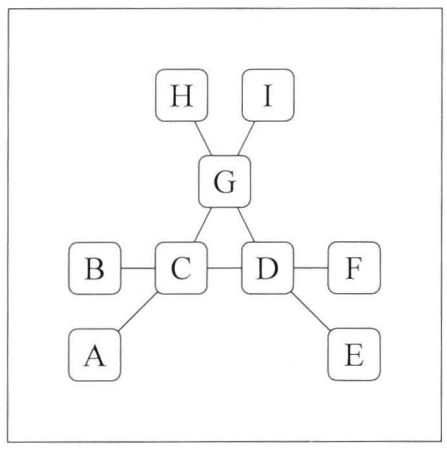
3. If a vertex appears in two clusters C_i and C_j, it must also appear in every cluster C_h on the path connecting vertices i and j in the junction tree. This is known as the *running intersection property*.

The *separator* of edge i-j in a junction tree is defined as $C_i \cap C_j$. The *width* of a junction tree is the size of its largest cluster minus one.

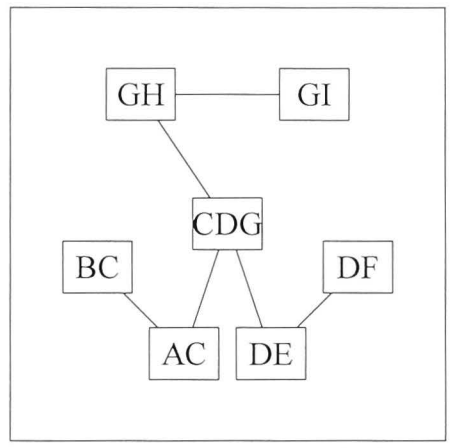
One of the steps in obtaining a junction tree from a graph is to triangulate the graph, *i.e.* add extra links such that every cycle of length greater than three has a chord. There are different ways to triangulate a graph, yielding different sets of clusters. Moreover, each triangulated graph may have several different junction trees. It is therefore desirable to have optimal triangulations and optimal junction trees with respect to complexity. As discussed later, the complexity here depends on the size of the clusters: an optimal junction tree minimises the size of the largest cluster. However, the optimality problem for triangulation is NP-complete. Given a triangulated graph, we can obtain an optimal junction tree using an algorithm from [53] which is quadratic in the number of cliques.

Figure 4.4 gives an example of three graphs and their junction trees. Note that the ith junction tree is also a junction tree for the jth graph if $i > j$ while it is not true if $i < j$. The 'best' junction tree is considered to be the first one (JT1) where the notion of 'best' is not only in relation to the tree width but also to the size of most of its clusters. The smaller the tree width or the cluster size, the better is considered the junction tree obtained. The biggest cluster for the first junction tree contains three elements against five for the last; furthermore, the first and second junction tree have the same largest cluster CDG, but the second largest cluster of the first junction tree is smaller than that of the second junction tree.

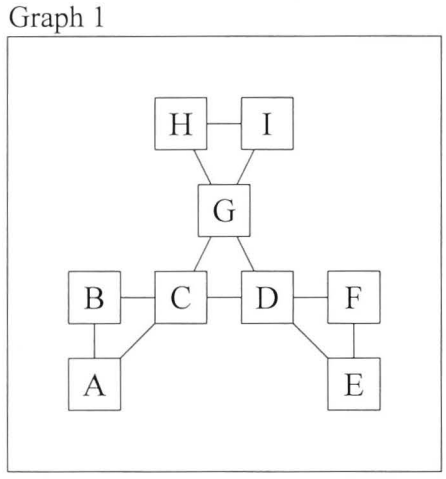
The reasoning behind the use of junction trees in diagnosis is that it could help avoid the need to compute a global diagnosis. Using a junction tree representation of a system has two main advantages [96]:



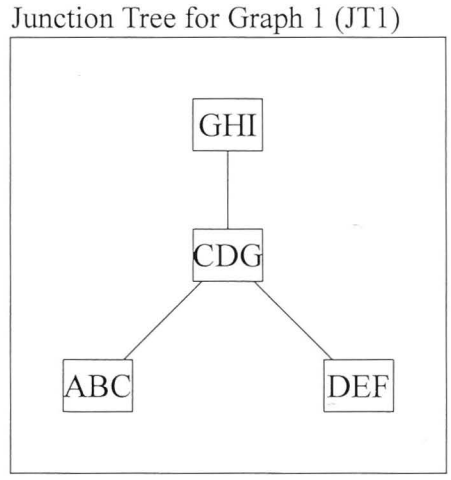
Graph 1



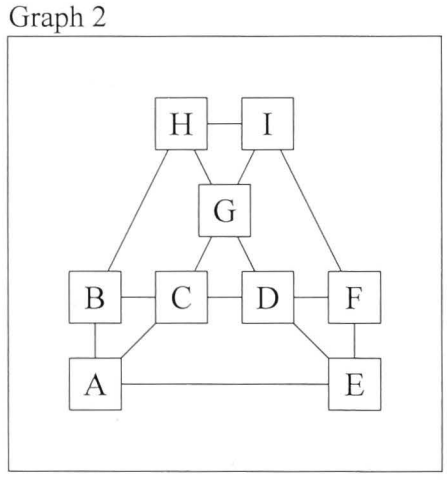
Junction Tree for Graph 1 (JT1)



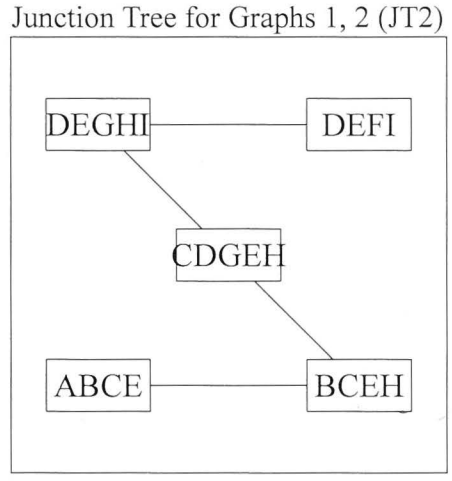
Graph 2



Junction Tree for Graphs 1, 2 (JT2)



Graph 3



Junction Tree for Graphs 1, 2, 3 (JT3)

Figure 4.4: Three graphs and corresponding junction trees

1. A tree representation of a system implies that local consistency is equivalent to global consistency.
2. Non-termination issues with local consistency algorithms can be resolved.

4.3.2 Distribution Algorithm

The junction tree algorithm returns a topology as defined previously, provided it is followed by computation of the edges of the tree itself. Indeed, let $e \in \Sigma_{S_1} \cap \Sigma_{S_2}$ be an event that is shared by subsystems S_i and S_j. We prove that any vertex S in the path between S_i and S_j contains this event ($e \in \Sigma_S$). There are two (possibly identical) components γ_1 and γ_2 such that $\forall i \in \{1,2\}$, $e \in \Sigma_i$ and $\gamma_i \in \mathcal{S}_i$. Since component γ_1 and γ_2 share an event, they are connected in the original topology and because of the second property of junction trees, there is a cluster S in the junction tree that contains both components ($\{\gamma_1, \gamma_2\} \subseteq S$). By the third property of the junction tree, all clusters between S_i and S contain component γ_i and thus event e. S can be between S_i and S_j or

outside, but in both cases there is a path of clusters between S_1 and S_2 that share event e. Thus, the junction tree algorithm returns a tree-shaped distribution. □

Algorithm 2 Distribution using Junction Tree Algorithm

1: **input** $\Gamma, \{\text{Mod}_1, \ldots, \text{Mod}_n\}$
2: $\mathcal{V} := \Gamma$
3: $\mathcal{E} := \{\langle ver_i, ver_j \rangle \in \mathcal{V}^2 \mid i \neq j \;\&\; \Sigma_i \cap \Sigma_j \neq \emptyset\}$
4: $\mathcal{S} := \{\}$
5: **while** $\mathcal{V} \neq \emptyset$ **do**
6: pick a vertex $ver \in \mathcal{V}$
7: $C := \{ver\} \cup \{ver' \mid \langle ver, ver' \rangle \in \mathcal{E}\}$
8: $\mathcal{E} := \mathcal{E} \cup \{\langle ver_1, ver_2 \rangle \mid ver_1 \in C, ver_2 \in C\}$
9: $\mathcal{V} := \mathcal{V} - \{ver\}$
10: $\mathcal{E} := \mathcal{E} - \{\langle ver_1, ver_2 \rangle \in \mathcal{E} \mid ver_1 = ver \vee ver_2 = ver\}$
11: **if** not $(\exists C' \in \mathcal{S} \mid C \subseteq C')$ **then**
12: $\mathcal{S} := \mathcal{S} \cup \{C\}$
13: **return** $\mathcal{S}$

We perform distribution by rearranging the topology of the system into a junction tree, as described in Algorithm 2. We first obtain a graph of the original system, $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Each component γ in the system is a vertex $ver \in \mathcal{V}$ on the graph. The edges, $edg \in \mathcal{E}$, on the graph represent connected components. We use the junction tree algorithm [49] to obtain the clusters that make up $\mathcal{S}$ (in iterative steps 6 to 13). This can be described as follows. We pick a vertex $ver \in \mathcal{V}$. A cluster C is obtained by taking the set formed by *ver* and its neighbours, *i.e.* the vertices on the graph that are connected to *ver* by an edge. We add edges so that all the vertices that make up a cluster are connected. C is added to $\mathcal{S}$ if it is not a subset of an element of $\mathcal{S}$. We update the original graph by removing *ver* and its associated edges from it. This procedure is repeated until no more vertices are left on the original graph. It is then trivial to calculate the separators that link the clusters into a junction tree.

As mentioned, building an optimal junction tree is NP-complete. However, we can use heuristics in the vertex selection phase of the algorithm (line 6) that would generally achieve polynomial-time while still producing a high quality tree [49]. One heuristic is to minimise the number of edges added to the graph [62] (line 8 of the algorithm), which then achieves a low-polynomial complexity.

Junction trees have been widely studied in various fields such as Constraint Satisfaction Problems (CSP) and Bayesian Network Inference (*e.g.* [100], [21]). This is because many NP-complete combinatorial problems on graphs are solvable in polynomial time when restricted to graphs of bounded tree width. The junction tree decomposition algorithm can be used to put a bound on tree width.

We mentioned in section 4.2.4 that local consistencies can be performed in any order. However, we can use a strategy, *global propagation* [49], that would only require two ordered series of local consistency computations on the junction tree to achieve global consistency. We assume a message pass from a cluster C_X to its neighbour C_Y to be an operation that makes the components of C_X locally consistent with those of C_Y. By performing these message passes in an ordered manner, we ensure that the consistency introduced by previous message passes is preserved.

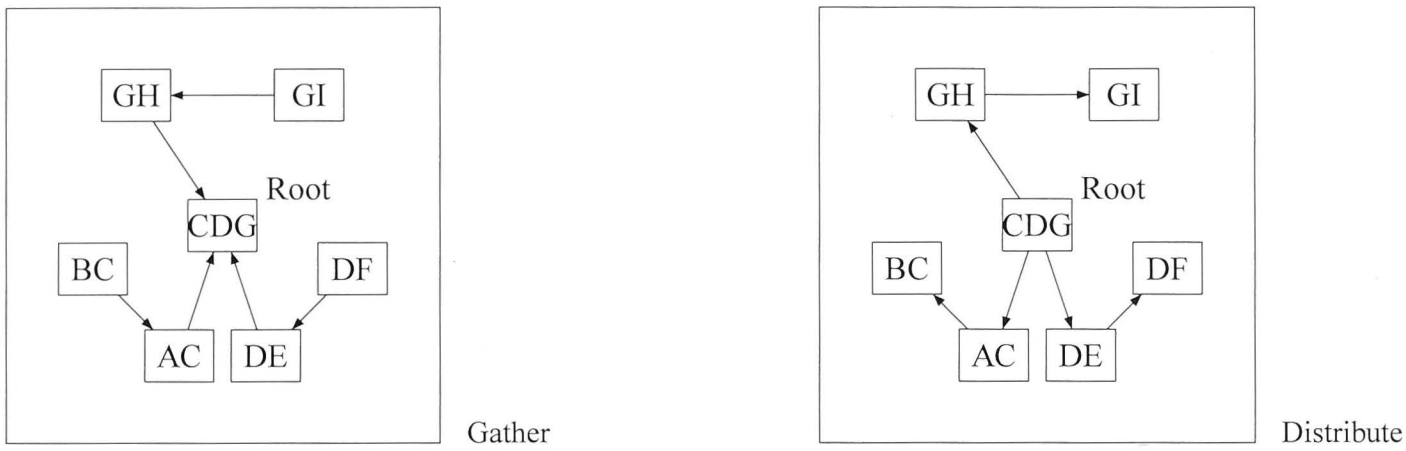


Figure 4.5: Global Propagation on Junction Tree

We arbitrarily pick a cluster $S_r \in \mathcal{S}$ to be the root of the junction tree. We start from each leaf node and perform local consistency with the neighbour until the root is reached (the *gather* phase). We then perform local consistency in the other direction, from the root back to the leaves (the *distribute* phase). All the clusters are now locally, and consequently globally, consistent with one another. This procedure is illustrated in Figure 4.5.

We implemented the distributed diagnosis algorithm 1 in **Java** and used algorithm 2 to generate the *distribution* (set of subsets) $\mathcal{S}$. We tested the performance of our proposed distributed algorithm on a variety of systems. More details and results are presented in section 4.5.

In addition to comparing diagnostic performance on different types of systems, we also compared diagnostic performance of our proposed algorithm to another distributed diagnosis algorithm in the literature by the authors of [96]. Results of this comparison are also presented in section 4.5.

4.4 Networks

A network in the physical sense is a set of interconnected components. The most flagrant example is the Internet where computers are linked together through telecommunication channels. The electricity grid is also another example of an important network that connects generators, transformers and loads. At the abstract level, a network is a *graph* in the mathematical sense.

The components of a network are represented as the *vertices* of a graph (as defined in Definition 28 and they are connected by *edges* according to some defined rules. A network has an associated physical *topology* (structure), which describes which components is connected to which other components. In the networks we deal with, it is assumed that no self-loop exist and that

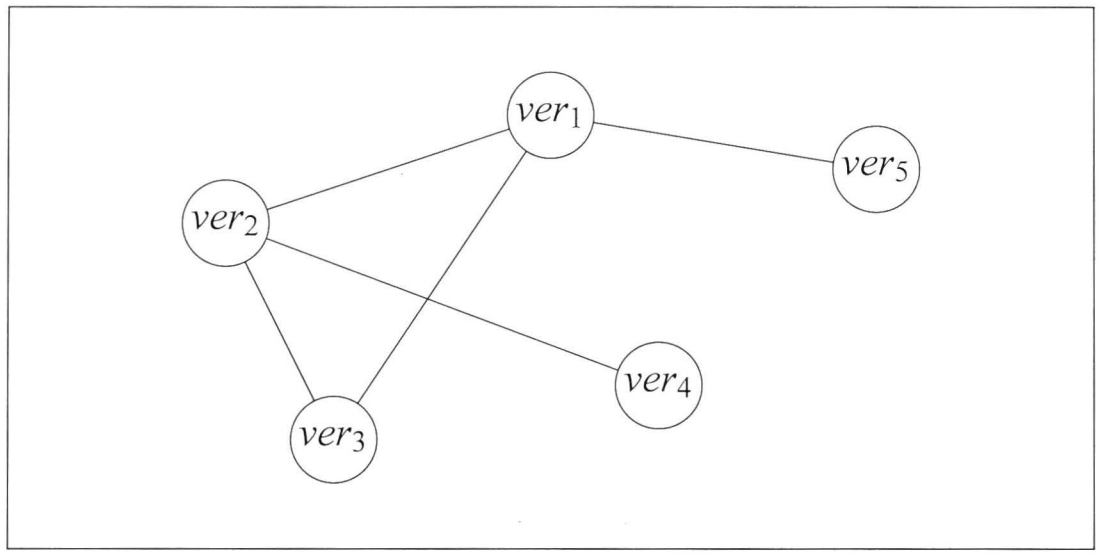


Figure 4.6: Example of a network with 5 vertices and 5 edges

the representative graphs are undirected with no repeated edge. An example of a graph with five vertices is shown in figure 4.6.

The junction tree provides a method to re-arrange the topology of a network on an abstracted level, without having to change anything in its actual physical configuration; the motivation being to make analysis easier.

4.4.1 Network Metrics

To compare inherent characteristics of various types of networks, we need a set of metrics that is able to capture and quantify certain properties that are displayed by the networks. We describe three such metrics in this section.

4.4.1.1 Average Path Length

To explain what is meant by *average path length*, we first need to define the concept of *distance* between two vertices.

Definition 31 (Distance between two vertices)**.** The *distance* d_{ij} between two vertices labelled ver_i and ver_j is given by the total number of edges that connect them through shortest linkages.

For example, for the graph represented in figure 4.6, the distance d_{14} between vertices ver_1 and ver_2 is equal to 2. Similarly, $d_{12} = 1$ and $d_{35} = 3$.

Definition 32 (Average Path Length)**.** The average path length of a network is defined to be the average value of all distances over the network:

$$\bar{\mathrm{L}} = \frac{2}{N(N-1)} \sum_{i,j(i<j)} d_{ij} \tag{4.3}$$

where N is the *size* of the network, *i.e.* the total number of vertices in the network.

For the network shown in figure 4.6, the average path length is 1.6 given the respective distances between vertices as shown in table 4.1 :

Value	d_{ij}
1	$d_{12}, d_{13}, d_{23}, d_{24}, d_{15}$
2	$d_{14}, d_{34}, d_{25}, d_{35}$
3	d_{45}

Table 4.1: distances in example network

The average path length of a network gives an idea of how easily nodes can be reached in the network. Generally, a short average path length is more desirable as it means information passing between nodes can be done faster.

4.4.1.2 Clustering Coefficient

The *clustering coefficient* of a network gives an idea of how isolated or connected the network is.

Definition 33 (Clustering Coefficient)**.** Let ver_i be a vertex in a network, where ver_i has k_i edges connecting it to k_i other vertices known as the *neighbours* of ver_i. The maximum number of edges among those k_i vertices is given by $k_i(k_i - 1)/2$. Let K_i be the actual number of edges existing between the k_i vertices. The *clustering coefficient* C_i of vertex ver_i is given by

$$C_i = \frac{2K_i}{k_i(k_i - 1)} \tag{4.4}$$

The *clustering coefficient* of the whole network is the averaged value of the clustering coefficients of all the vertices in the network ($0 \leq C \leq 1$). $C = 0$ if and only if all vertices in the network are isolated, *i.e.* have neighbours that are not connected to each other, and $C = 1$ if each vertex is connected to every other vertex in the network. The clustering coefficient is a measure of how likely it is that the neighbours of a node in the network are connected to each other.

We once again consider the example network in figure 4.6 to illustrate. Table 4.2 shows the clustering coefficient values for the vertices shown in figure 4.6.

Vertex	**Number of neighbours**	K_i	C_i
ver_1	3	1	$\frac{1}{3}$
ver_2	3	1	$\frac{1}{3}$
ver_3	2	1	1
ver_4	1	0	0
ver_5	1	0	0

Table 4.2: clustering coefficients in example network

Hence, the clustering coefficient C of the network, given by the average of all clustering coefficients in table 4.2 is equal to $(\frac{1}{3} + \frac{1}{3} + 1 + 0 + 0)/5 = \frac{1}{3}$.

4.4.1.3 Degree and Degree Distribution

We define here the concept of the *degree* of a vertex in an *undirected* network.

Definition 34 (Degree)**.** The degree of a vertex ver_i in an undirected network is the number k_i of the edges connecting it to its k_i neighbours.

In a *directed* network, we distinguish between incoming edges and outgoing edges of a node. But for the undirected network case, we do not have to worry about that.

Intuitively, a vertex of higher degree will have more significant influence on the network because it is involved in more connections.

The *average degree* $\langle k \rangle$ of a network is the average value of vertex degrees over the entire network.

In a network, every vertex has a degree value, some large and some small. The distribution of vertices of certain degree could be of interest to better understand the network. This distribution is called the *degree distribution* of the network.

Definition 35 (Degree Distribution)**.** The degree distribution of a network is defined by a probability function $Prob\,(k)$, which is the probability that a randomly picked vertex will have degree k, assuming each vertex has equal probability to be picked (uniform distribution).

Once more, we use the example given in figure 4.6 to illustrate. The degrees of each vertex is equivalent to its number of neighbours and hence is given by column 2 in table 4.2. The average degree $\langle k \rangle$ of the network is equal to $(\frac{3+3+2+1+1}{5}) = 2$.

k	$Prob\,(k)$
1	2/5
2	1/5
3	2/5

Table 4.3: degree distribution in example network

4.4.2 Network Configurations

Next, we describe a few different types of physical topology for networks. These topologies are used as starting structures for systems on which we test the performance of our distributed diagnosis algorithm. Understanding the relationship between the topology of networks and the performance of our algorithm provides insight into how best to design networks or how best to tackle operations on networks given their structures.

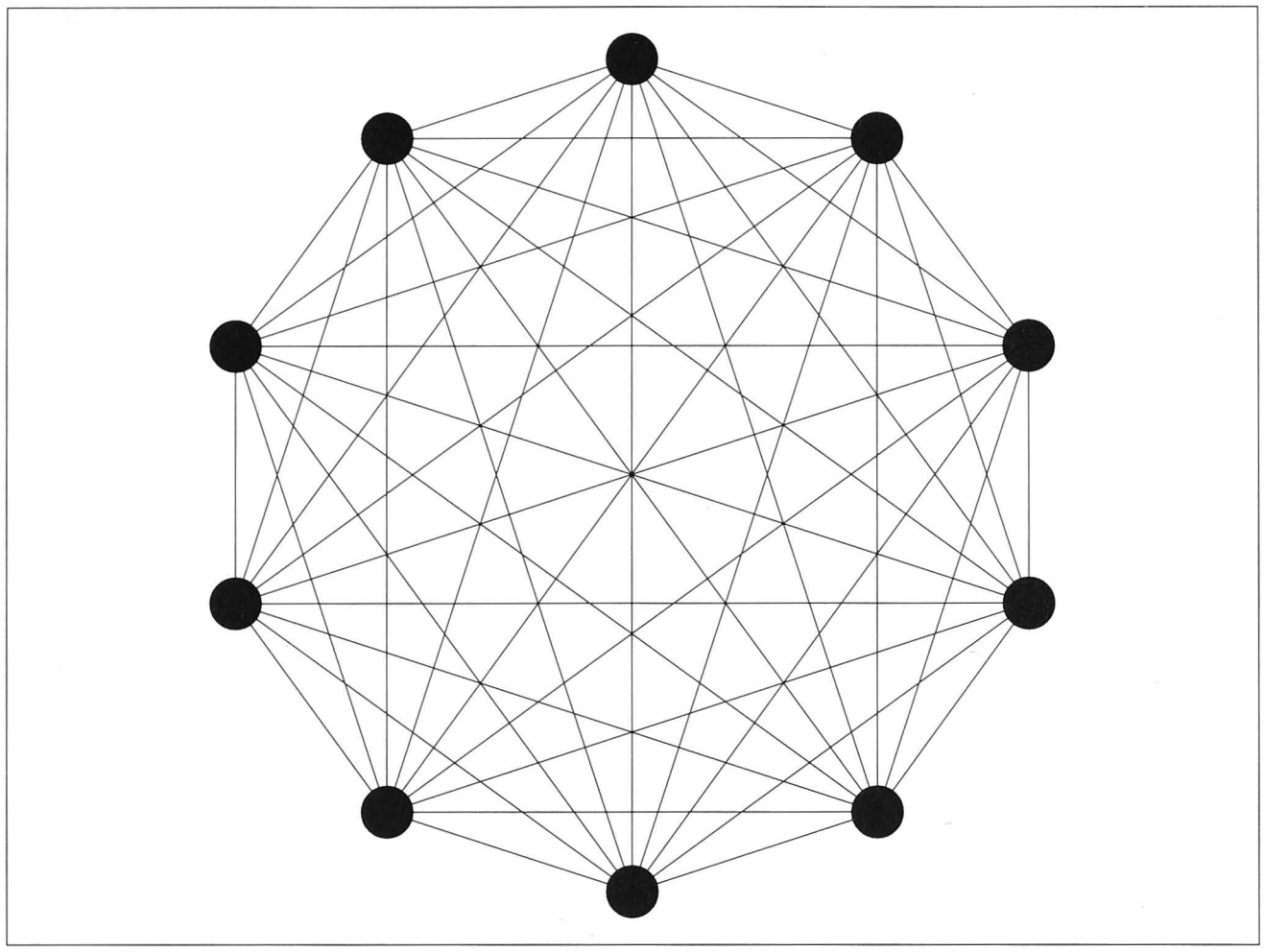


Figure 4.7: Example of a fully connected network with 10 vertices

We explore three types of *regular* networks (fully connected, branching and ring networks) and two types of randomised networks (random networks and small world networks). *Regular* networks are constructed in a predictable way whereas randomised networks have an element of randomness in their structure.

4.4.2.1 Fully Connected Network

A *fully connected* network is a type of regular network where there exists an edge between any pair of vertices in the network. An example of a fully connected network is shown in figure 4.7.

A fully connected network has an average path length given by $\overline{L}_{full} = 1$ and a clustering coefficient given by $C_{full} = 1$, both values being self-evident from their respective definitions 33 and 32. The total number of edges in a fully connected network of size N (*i.e.* with N vertices) is $\frac{N(N-1)}{2}$ [22]. We note that among all networks with the same number of vertices, the fully connected network has the shortest average path length of 1, and the highest clustering coefficient of 1.

4.4.2.2 Ring Network

In a ring network of size N, each node is connected to $2K$ nearest-neighbours, where $K > 0$ is an integer. A ring network with $K = 2$ is shown in figure 4.8.

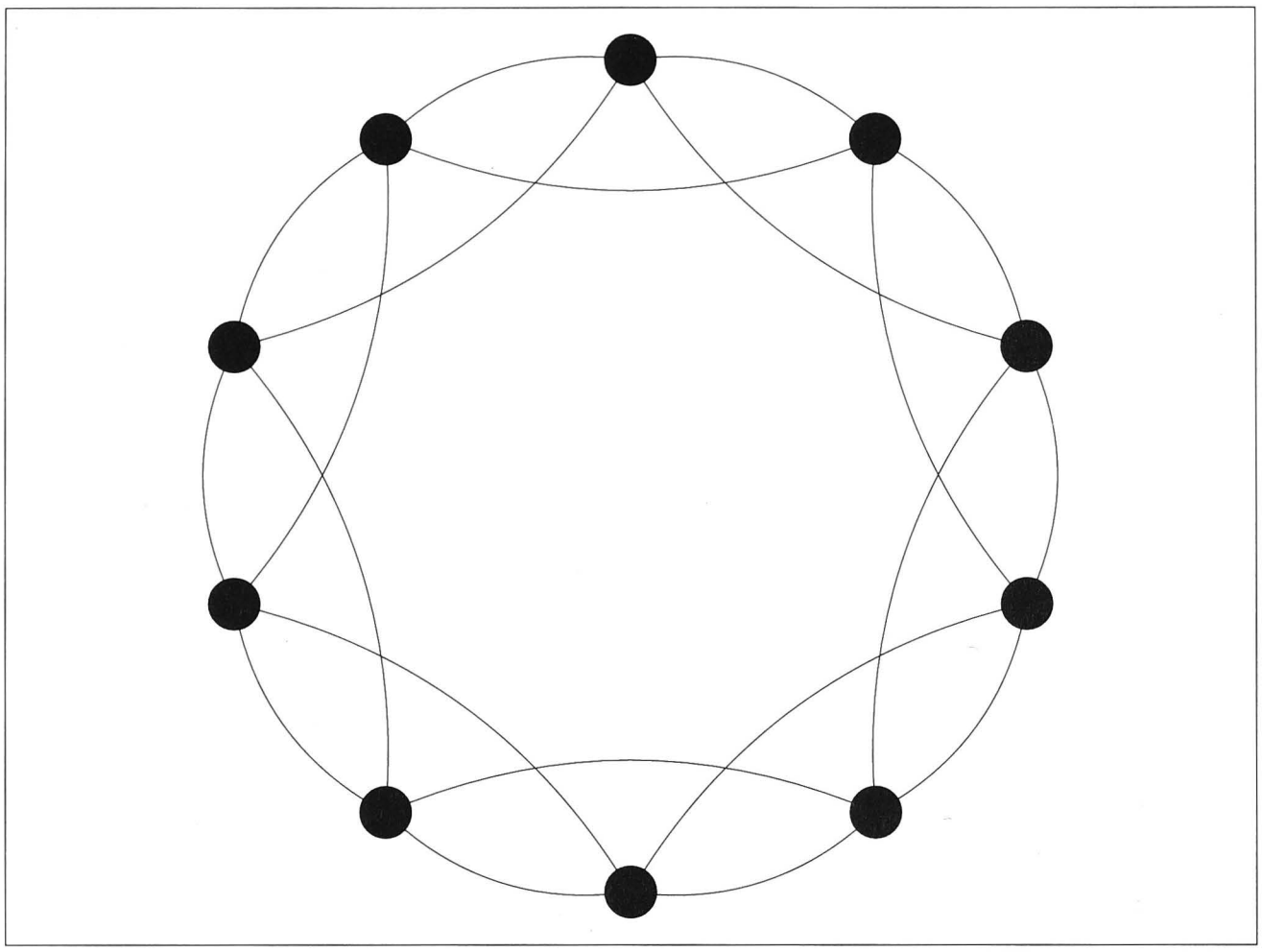


Figure 4.8: Example of a ring network with ten vertices and $K = 2$

The clustering coefficient for a ring network is given by

$$C_{ring} = \frac{3(K-1)}{2(2K-1)} \tag{4.5}$$

For large K ($K \to \infty$), $C_{ring} \to \frac{3}{4}$.

The average path length of a ring network, given M number of edges in the network, is given by

$$\overline{L}_{ring} = \frac{M(M+1) - 2(K-1)(M-K+1)}{2M} \tag{4.6}$$

For large M ($M \to \infty$), $\overline{L}_{ring} \to \infty$. Derivations for C_{ring} and $\overline{L}_{ring}$ can be found in [22].

4.4.2.3 Branching Network

A branching network has a tree structure where the vertices of the tree are the actual nodes of the network. It can further be described as an *r-ary* tree whereby each vertex which is not a leaf has exactly r children. A 2-ary tree is more commonly known as a binary tree and a 3-ary tree as a ternary tree. We consider the root of the tree to be of *order* 0, and we write $\rho = 0$. The *tree order* is the partial ordering on the vertices of a tree with $u \leq v$ if and only if the unique path from the root to v passes through u. The leaf vertices are considered to be at layer η ($\rho = q$). Examples of a branching networks are shown in figure 4.9 and 4.10.

The clustering coefficient for a branching network is given by $C_{branch} = 0$, self-evident from

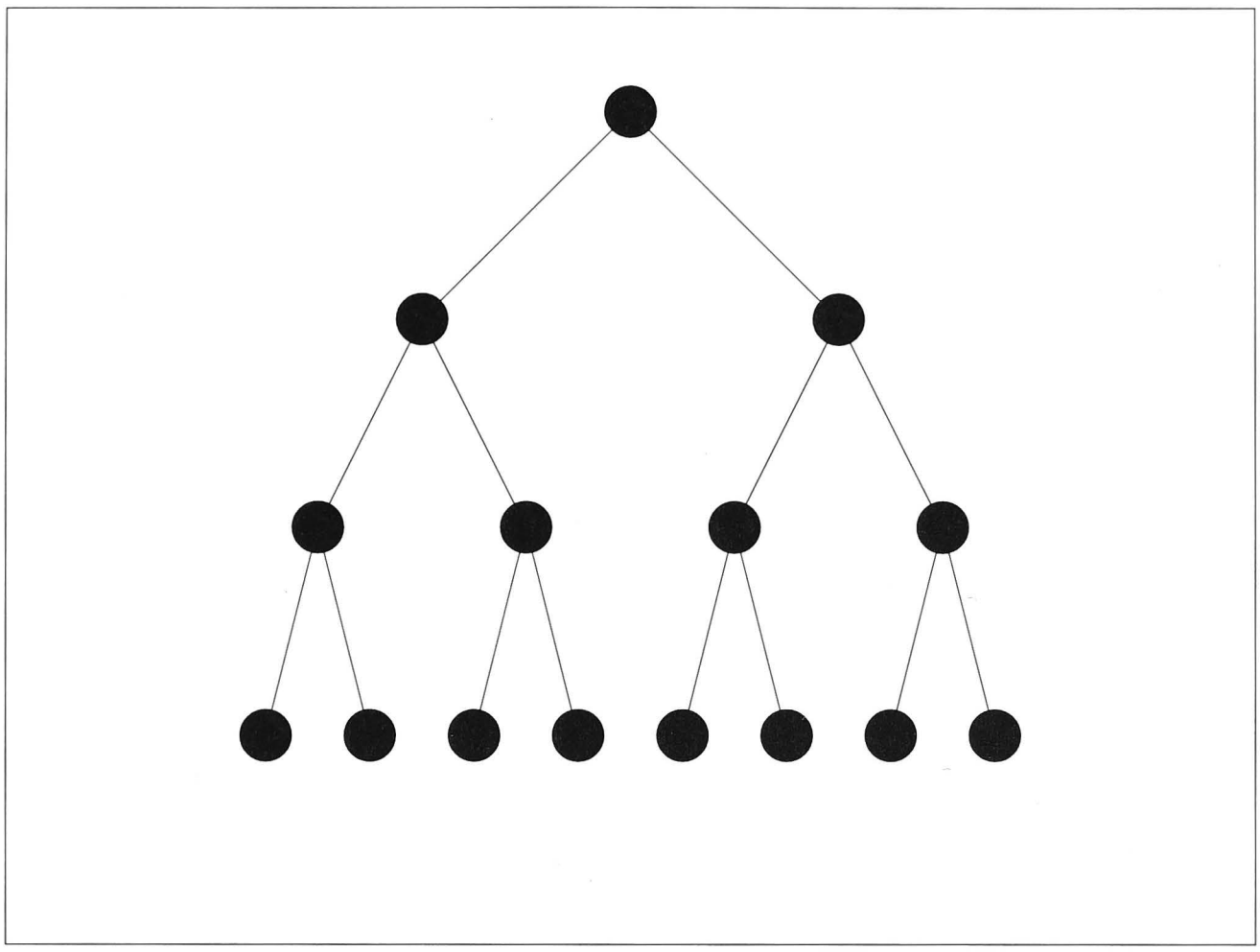


Figure 4.9: Example of a branching network with $n = 3$ and $r = 2$

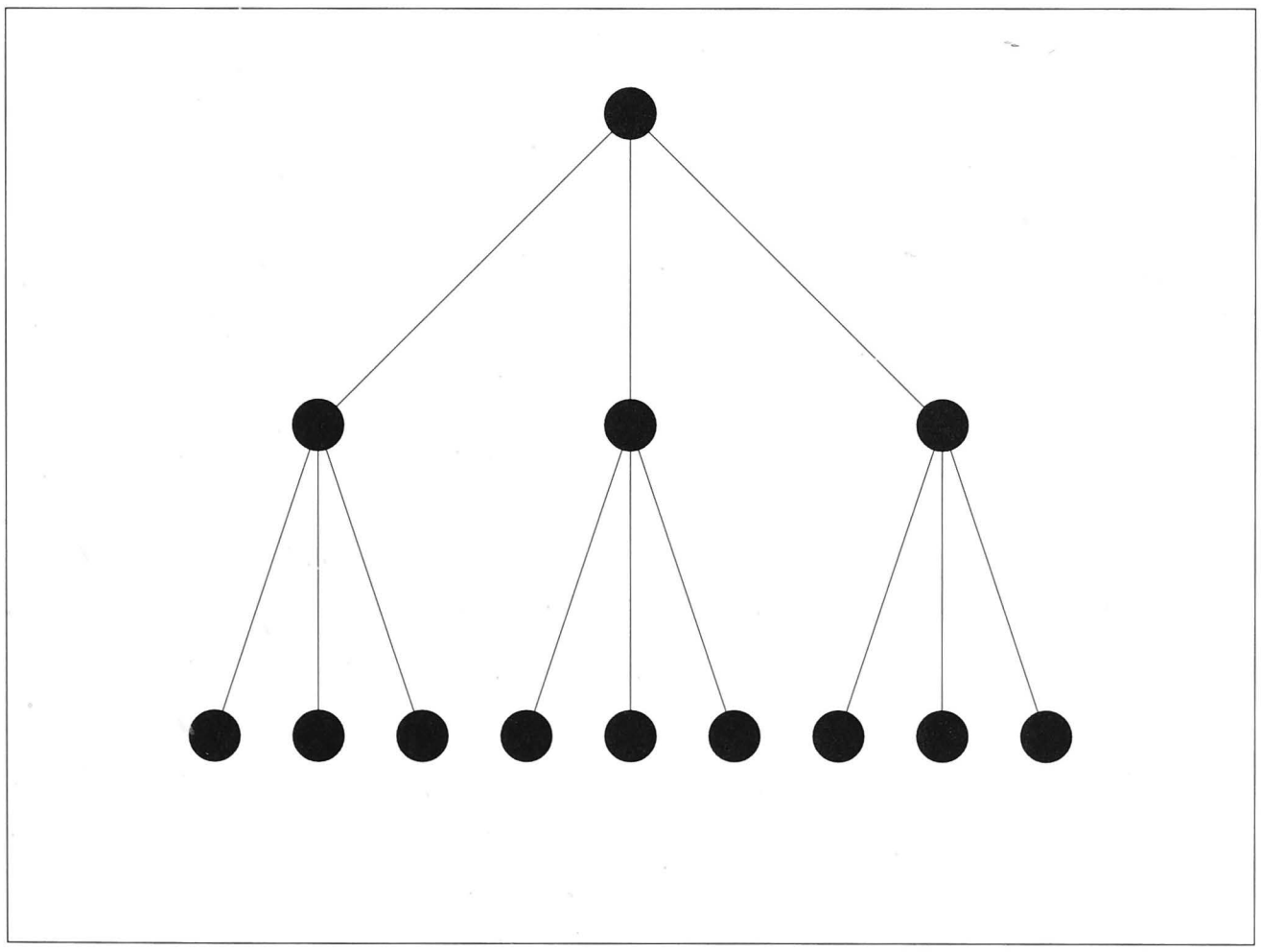


Figure 4.10: Example of a branching network with $n = 2$ and $r = 3$

definition 33.

Let ρ represent a layer in a branching network. The number of edges found in layer ρ is given by $r^{\rho+1}$. The average path length for a branching network is given by

$$\overline{L}_{branch} = \frac{1}{(r-1)(r^{q+1}-1)} \sum_{\rho=0}^{q-1} r^{\rho+1} \left(r^{q-\rho}-1\right)\left(r^{q+1}-r^{q-\rho}\right) \tag{4.7}$$

The proof for equation 4.7 is derived as follows. *

Proof. The average path length in a branching network with q layers and r children per node is given by the total path length of the network divided by the total number of components in the network. The total path length is given by the sum of distances between any two vertices in the tree. To obtain the number of times a given edge is traversed from layer ρ to $\rho + 1$, we imagine deleting the edge so that the tree is divided into two sub-trees. The number of times the edge is counted is given by the product of the number of nodes in each of the sub-trees:

$$\left(\frac{r^{q-\rho}-1}{r-1}\right)\left(\frac{r^{q+1}-r^{q-\rho}}{r-1}\right) \tag{4.8}$$

Hence, as there are $r^{\rho+1}$ edges in a layer, the total path length of the network is given by:

$$\sum_{\rho=0}^{q-1} r^{\rho+1} \left(\frac{r^{q-\rho}-1}{r-1}\right)\left(\frac{r^{q+1}-r^{q-\rho}}{r-1}\right) \tag{4.9}$$

The total number of components in the network is given by $\left(\frac{r^{q+1}-1}{r-1}\right)$ and therefore we divide the total path length by the total number of components to obtain the average path length $\overline{L}_{branch}$ of the network:

$$\overline{L}_{branch} = \frac{1}{(r-1)(r^{q+1}-1)} \sum_{\rho=0}^{q-1} r^{\rho+1} \left(r^{q-\rho}-1\right)\left(r^{q+1}-r^{q-\rho}\right)$$

which is equation 4.7. □

Figure 4.11 shows a plot of the average path length $\overline{L}_{branch}$ of a branching network against q for different values of r.

For $r > 1$, as $q \to \infty$, $\overline{L}_{branch} \to \infty$. $\overline{L}_{branch}$ increases exponentially with respect to q. From equation 4.7, it is interesting to note that for $q = 1$, *i.e.* for a star network, as $r \to \infty, \overline{L}_{branch} \to r$.

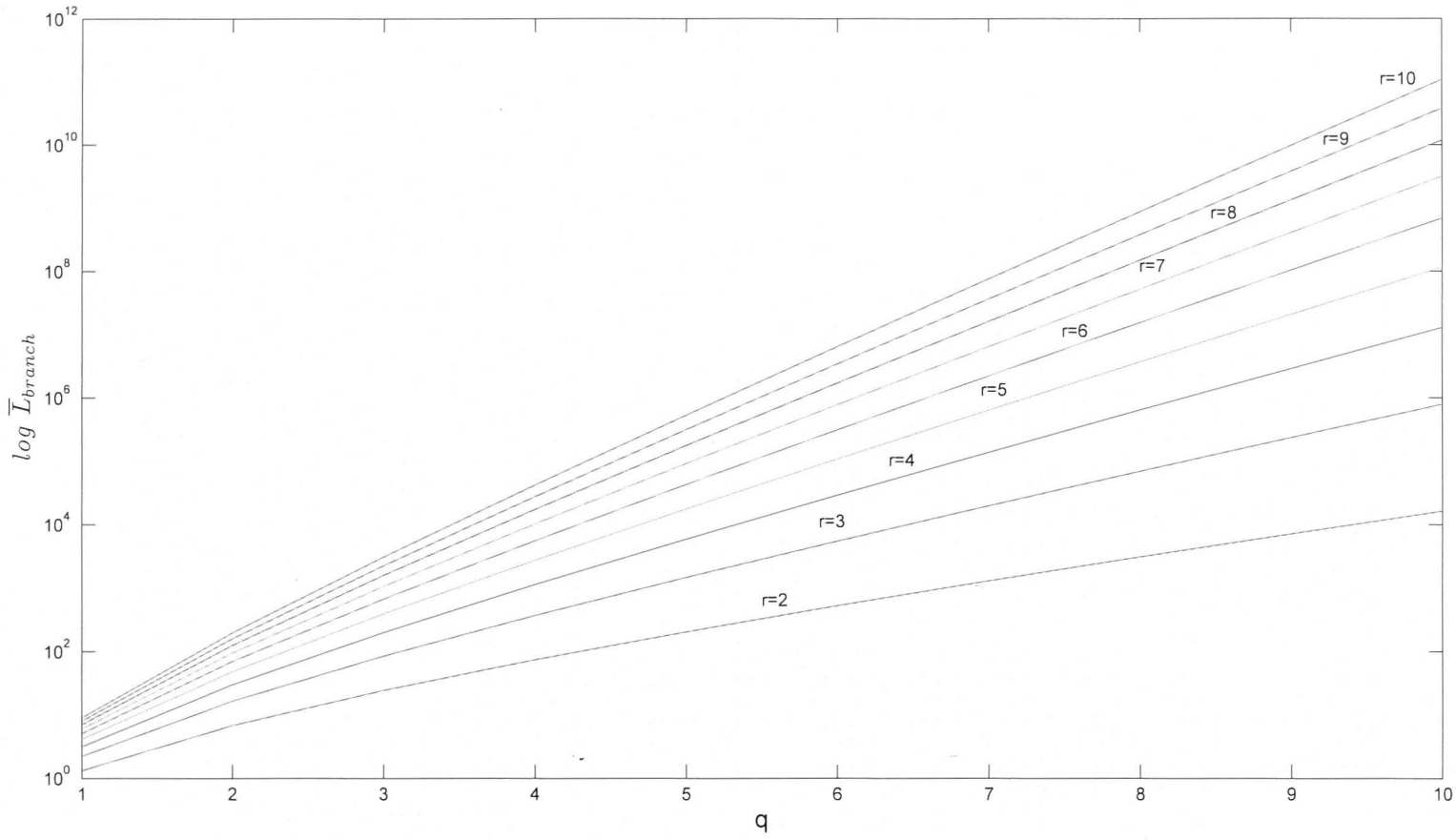


Figure 4.11: Plot of $\log \overline{L}_{branch}$ versus q for different values of r

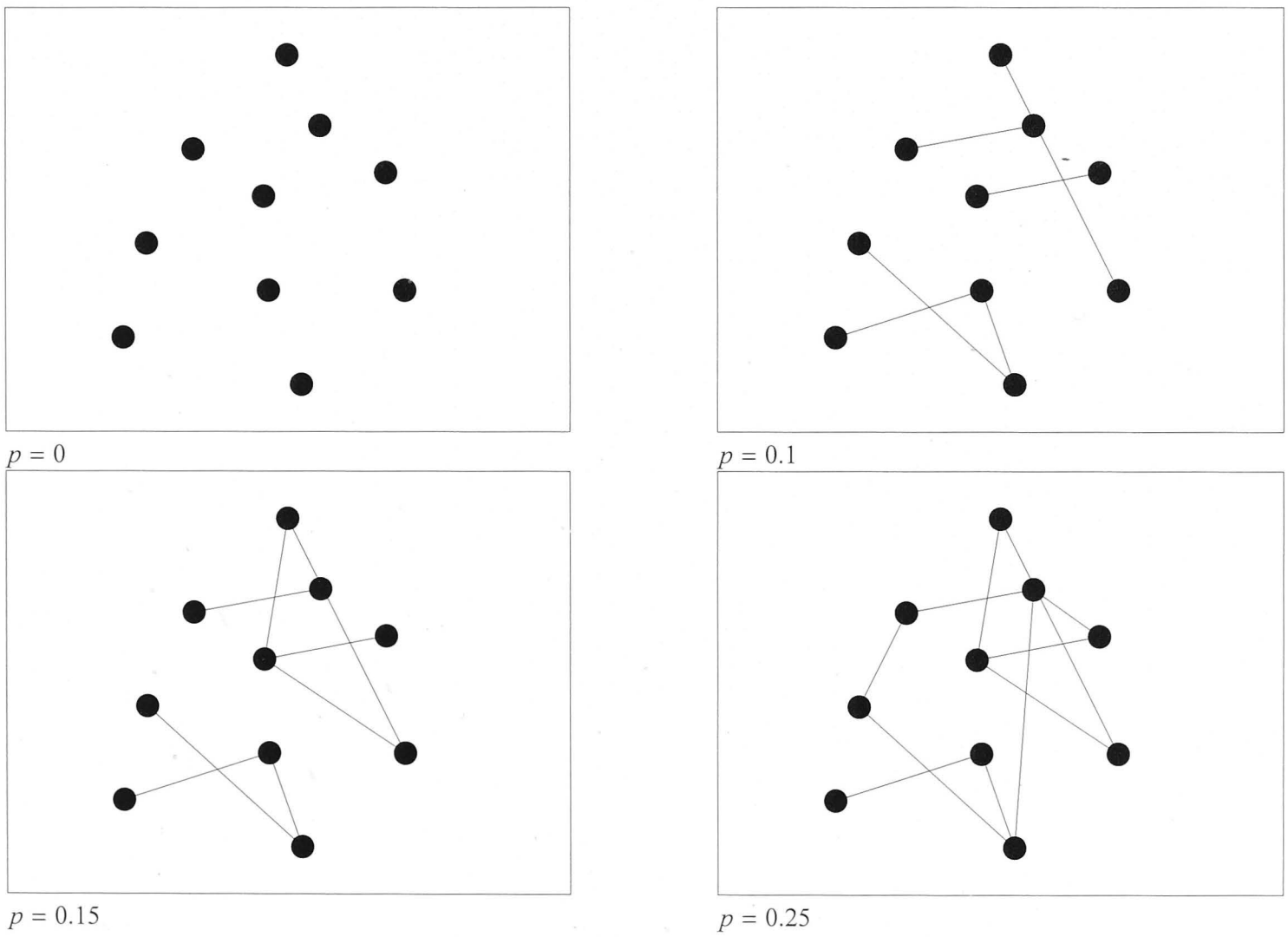


Figure 4.12: Example of random networks with 10 vertices as p is increased

4.4.2.4 Random Network

A random network is the opposite extreme of a regular network. There is no predetermined structure to the graph and vertices are connected with an element of randomness. The random network was introduced by Erdös and Rényi in their seminal papers [34, 35]. A random network consists of N vertices joined by edges which are placed between pairs of vertices chosen uniformly at random. Every possible edge between any two vertices is present with probability p from a uniform distribution, and absent with probability $1 - p$. To construct a random network, we start with N isolated vertices, where $N >> 1$. Two vertices are randomly picked and then connected by an edge with probability p from a uniform distribution. The vertices are then put back to the pool and the process is repeated for a suitably large number of steps. The procedure will yield approximately $pN(N-1)/2$ number of edges. Figure 4.12 illustrates some random graphs generated from 10 isolated vertices with different values of p.

A significant result in [35] is that important properties of random graphs emerge at certain values of p. For example, when p is larger than a certain threshold $p_c \sim \frac{\ln N}{N}$, almost all random graphs generated in the above described way will be connected, whereas for $p < p_c$, almost all graphs generated were not connected networks (*i.e.* contain isolated clusters).

The average degree of a random network of size N is given by

$$\langle k \rangle_{rand} = p(N-1) \tag{4.10}$$

The average path length satisfies

$$\overline{L}_{rand} \sim \frac{\ln N}{\ln \langle k \rangle_{rand}} \tag{4.11}$$

The clustering coefficient is given by

$$C_{rand} = p \approx \frac{\langle k \rangle_{rand}}{N} \tag{4.12}$$

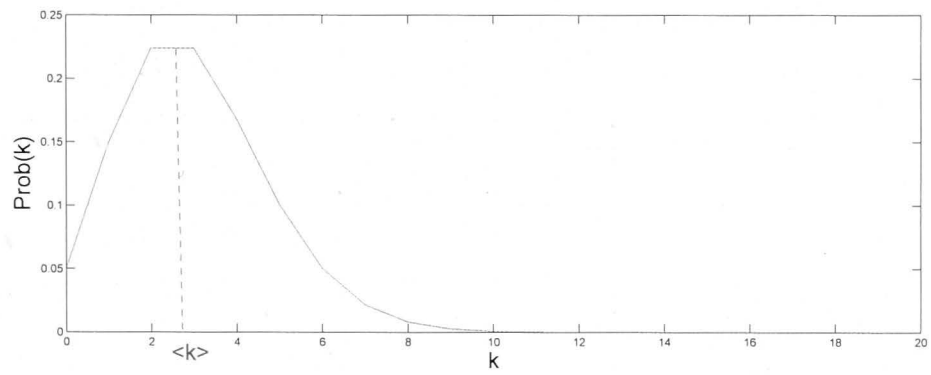


Figure 4.13: Degree distribution in a random network

*Derivation for the proof builds on a post in a discussion forum about total path length in a binary tree - http://www.mathkb.com/Uwe/Forum.aspx/math/54942/average-path-length

The degree distribution in a random network follows a *Poisson distribution*:

$$Prob\,(k) = \frac{\mu^k}{k!}\exp^{-\mu} \tag{4.13}$$

where μ is the expectation value, $\mu = pN \approx \langle \mathrm{k} \rangle_{\mathrm{rand}}$. An illustration of the degree distribution of a random network is shown in figure 4.13. The proofs for equations (4.10) to (4.12) can be found in [22].

We note that when N is large, such random networks may have a relatively small average path length because the growth of $\ln N$ is much slower than that of N in equation (4.11). Also from equation (4.12), a large scale random network (large N) does not have prominent clustering features.

4.4.2.5 Small-World Network

As mentioned previously, a ring network has a high clustering coefficient and a large average path length. On the other hand, a random network possesses a small clustering coefficient and displays short average path length. There is a category of networks that is characterised by both a high clustering coefficient and a short average path length. This category is known as the *small-world* networks. They were introduced by Watts and Strogatz in [98]. Since then, there have been modified versions of the algorithm presented in [98] but we will stick to the original version in this thesis when referring to *small-world* networks.

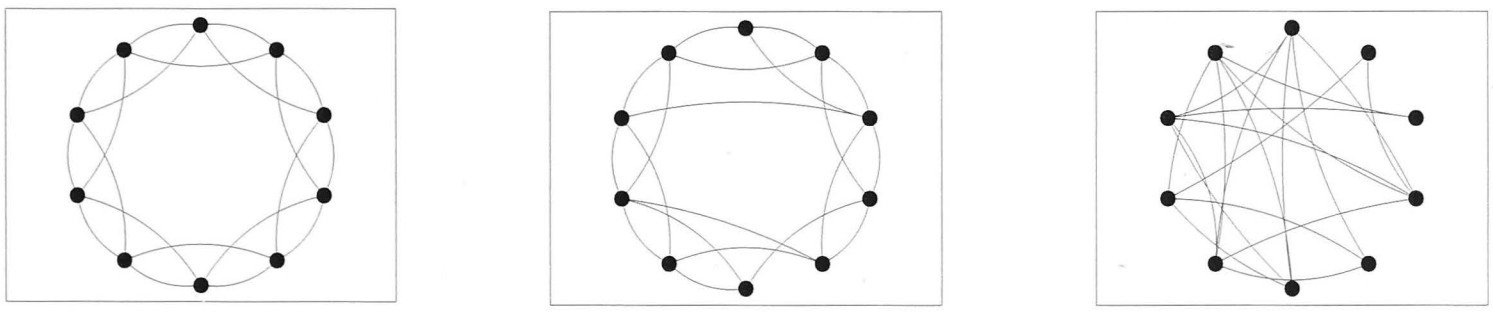


Figure 4.14: Transition from regular to small-world to random networks as the value of p increases

A small-world network can be generated as follows [98]:

1. We start with a ring network with N vertices, each connected to $2K$ nearest neighbours.
2. We choose a vertex and the edge that connects it to a nearest neighbour in a clockwise (or anti-clockwise) sense.
3. This edge is reconnected with probability p_{sw} to a vertex chosen uniformly at random over the entire ring, with duplicate edges forbidden; otherwise we leave the edge alone (with probability $1 - p_{sw}$.
4. This process is repeated by moving clockwise (or anti-clockwise) around the ring, considering each vertex in turn until one lap is completed.

5. We then consider edges that connect vertices to their second nearest neighbours clockwise (or anti-clockwise). These edges are also rewired as before with probability p_{sw}.

6. We continue to circulate around the ring and proceed outward to more distant neighbours after each lap until every edge in the network has been considered once.

Since there are NK edges in the network, the rewiring process stops after K laps. Three examples generated using this process, with different values of p_{sw}, are shown in figure 4.14. It can be observed that for the case where $p_{sw} = 0$, the original ring stays unchanged. As the value of p_{sw} increases, the graph becomes increasingly disordered until for $p_{sw} = 1$ where all edges are rewired randomly and a random network (as described in section 4.4.2.4) is obtained. One main result of [98] is that for $0 < p_{sw} < 1$, the graph displays *small-world* properties, *i.e.* high clustering yet small average path length.

For large enough size N, the clustering coefficient of the small-world network is given by

$$C_{sw}(p_{sw}) = \frac{3(K-1)}{2(2K-1)}(1-p_{sw})^3 \tag{4.14}$$

The average path length can be expressed as

$$\overline{L}_{sw}(p_{sw}) = \frac{2N}{K} f(2Np_{sw}/K) \tag{4.15}$$

where

$$f(x) = \begin{cases} c, & x << 1 \\ \frac{\ln x}{x} & x >> 1 \end{cases} \qquad \text{(typically } c = 1/4\text{)} \tag{4.16}$$

The proof for equations 4.14 and 4.15 can be found in [78].

4.5 Experimental Results

In this section, we present the results of the performance of our distributed diagnosis algorithm on a range of different graphs presented in section 4.4.2 as their size and other properties are varied.

We implemented Algorithm 2 in **Java** and compared it to another distributed diagnosis algorithm from *Su and Wonham* [96]. Experiments were run on an Intel Core 2D, 2.4 GHz, 2GB Linux machine.

The automaton model that was used for each component in a network had a total of $s+1$ states where s is the number of nearest neighbours of the component. The automaton consisted of a base state that can transition to a state corresponding to a connection with a nearest-neighbour. The total number of transitions is given by $2s$ where there is a transition e_i from the base state s_0 and a nearest-neighbour state s_i, and a corresponding return transition f_i between state s_i to the base state s_0. To illustrate, the automaton for a component that has three nearest neighbours is given in figure 4.15. A is the base state and has nearest neighbours B, C and D.

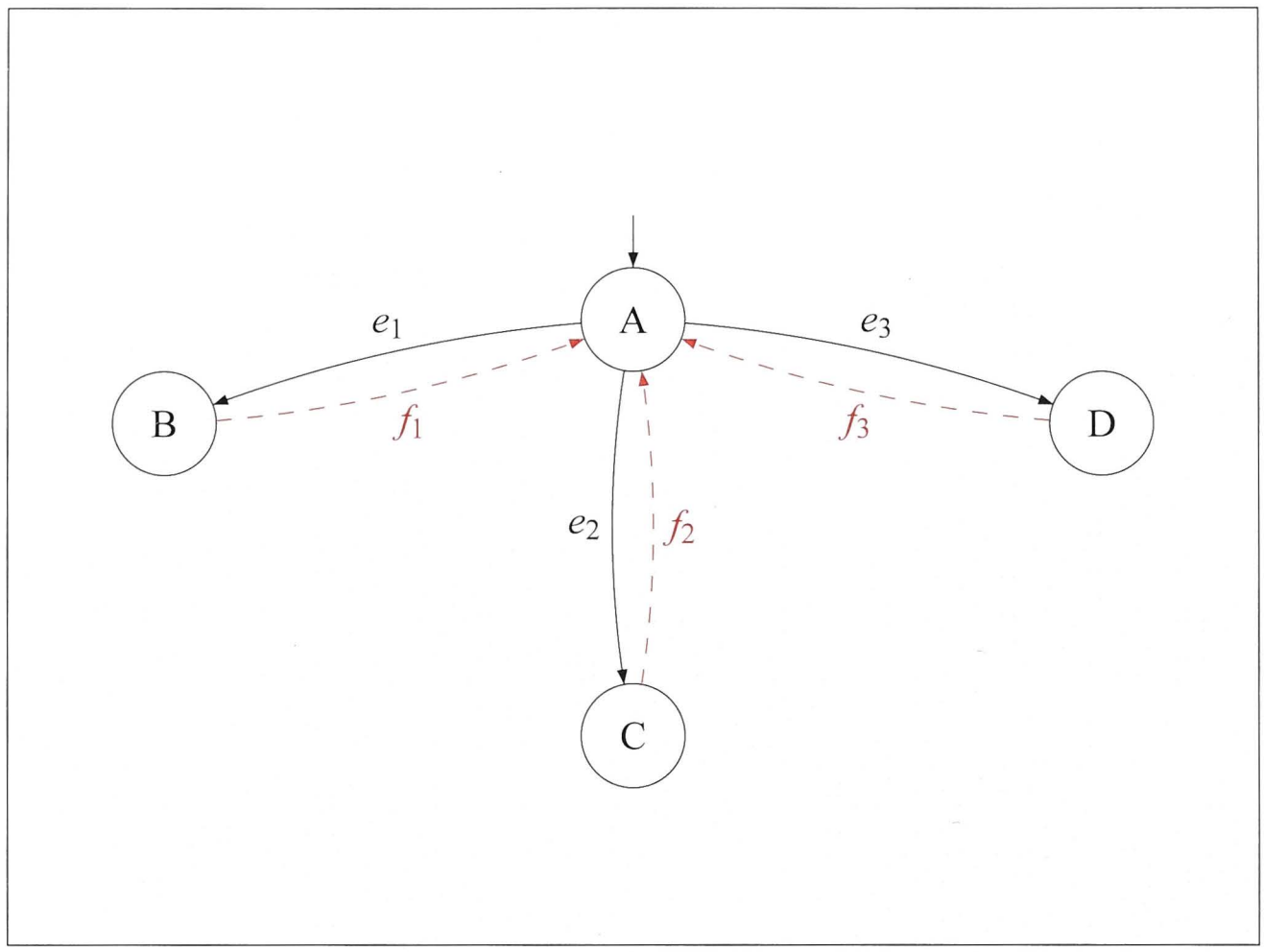


Figure 4.15: Test automaton representing a component having 3 nearest neighbours

4.5.1 Summary of Algorithm by Su and Wonham

Su and Wonham present a similar algorithm for the distributed diagnosis of DES in [96]. A local diagnosis is computed for each component. Then, a given diagnosis is incrementally synchronised with the other diagnoses, which ensures global consistency. After each synchronisation, the events that appear only in components that have already been synchronised can be safely abstracted: the current diagnosis is projected on the relevant events, which reduces the complexity. A heuristic ordering, based on minimising the state size of the synchronous product at each stage, is applied for synchronising the local diagnoses incrementally.

This algorithm can be seen as a special case of our approach with three main differences. First, it implicitly builds a *junction line* (*i.e.* a junction tree with only one leaf node), since the diagnoses are synchronised in sequence. This restriction potentially increases the width of the junction tree, with a negative impact on the global efficiency.

Second, this algorithm builds a junction tree/line on the graph of *events* rather than the graph of components. While this appears less intuitive, it can also be done in our approach. In this case, two events are connected in a graph of events iff they are shared by some component. Since all the events of a given component are interconnected, at least one cluster will contain all these events and will be initialised with the diagnosis of this component (potentially synchronised with the diagnosis of other components). Considering the graph of events leads to clusters with less, or in the worst case as many, events than in the approach presented in this chapter, thus reducing complexity.

Finally, a dynamic strategy to choose the order of the synchronisation is proposed by Su and Wonham, while in this thesis a junction tree is computed before the local diagnoses are generated and synchronised.

4.5.2 Results on Branching Networks

Table 4.4 shows results obtained on branch networks with varying parameter values for q and r. We note that a star network is a special case of a branching network where $q = 1$. The size of the largest structure ([number of nodes, number of edges] of the synchronisation automaton) and the time for producing a diagnostic result were recorded for each network for two cases: one using Su and Wonham's (S&W) algorithm, and two using our proposed junction tree (JT) approach. There are cases where memory is exhausted before the computation is finished and the largest structures reached are shown in red in table 4.4.

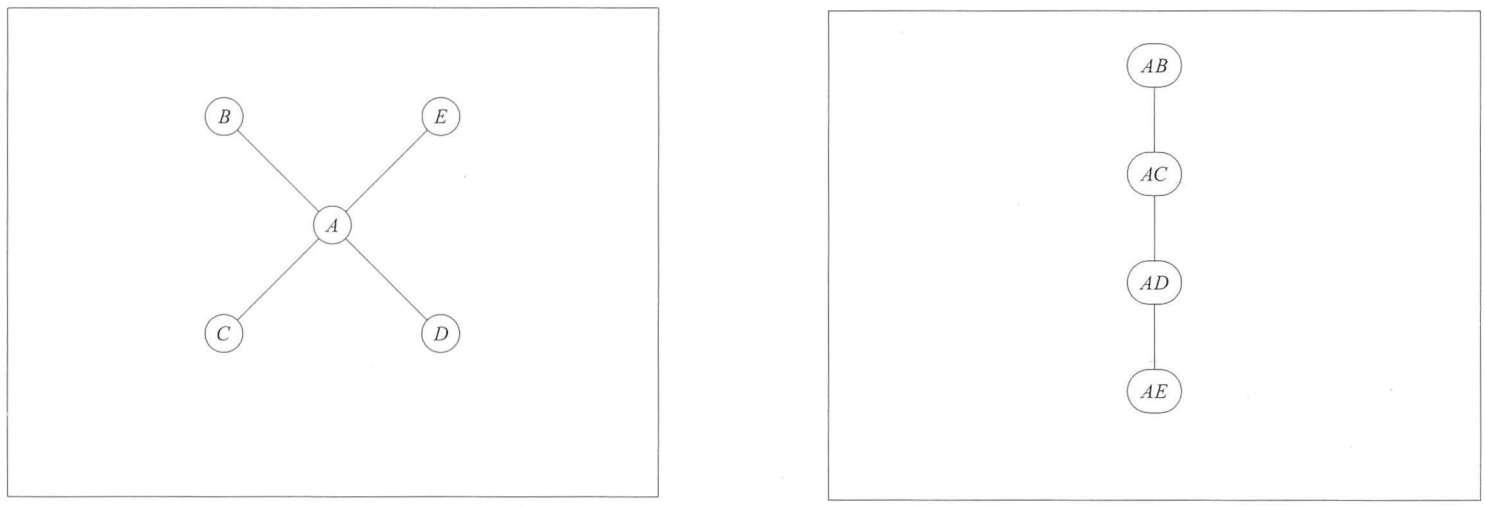
q	r	n	Su & Wonham		JT	
			largest struc	time (s)	largest struc	time (s)
1	5	6	[7, 12]	0.115	[37, 122]	0.103
1	10	11	[8, 24]	0.463	[122, 442]	0.220
1	15	16	[17, 32]	1.174	[257, 962]	0.340
1	20	21	[16, 64]	3.132	[442, 1682]	0.417
1	25	26	[16, 64]	3.053	[667, 2602]	0.898
1	30	31	[16, 64]	5.210	[962, 3722]	1.566
1	40	41	[32, 160]	17.914	[1682, 6562]	4.150
1	50	51	[32, 160]	44.367	[2602, 10202]	10.115
2	2	7	[4, 6]	0.127	[10, 26]	0.157
3	3	40	[96, 512]	7.339	[17, 50]	0.362
4	2	31	[54, 270]	5.097	[10, 26]	0.0914
4	3	121	[576, 4416]	1103.356	[17, 50]	1.324
4	5	781	[640, 5504]	na	[37, 122]	216.973
5	3	364	[256, 2048]	na	[17, 50]	18.835
6	2	127	[1458, 13122]	5820.996	[10,26]	1.415
7	2	255	[1296, 12096]	40488.092	[10,26]	6.148
8	2	511	[4374, 45198]	na	[10, 26]]	44.147

Table 4.4: Results on branching networks (where computation exhaust memory, the largest structures reached are shown in red)

A few interesting points emerge from the results on branching networks. Overall, the JT method performed better than S&W's method, especially when the network contains a large number of components. For networks containing around 10 components or less, the difference in the diagnosis time between the two methods is negligible although the largest structure in the JT method has a greater size.

For star networks ($q = 1$), the size of the largest structure is significantly larger in the JT method than in the S&W method despite that diagnosis time is better in the former. This can

be explained by the fact that in the S&W method, the 'junction line' is built on the graph of events rather than the graph of components and a heuristic ordering procedure is used to minimise the size of the synchronous automaton at every step. The star shape of the network also means that the root node appears in every cluster in the junction tree (figure 4.16), forcing multiple redundant synchronisation with the root node during diagnosis. Because the root node is the highest connected node, this creates large synchronising structures. The number of states in the largest synchronising structure in a star network is actually given by $n^2 + 1$.



Star Network Graph

Star Network Junction Tree

Figure 4.16: Example of a star network and its junction tree

Interestingly, for cases where $q > 1$, the size of the largest structure is more reasonable and depends on the value of the branching factor r only (and not on q). This makes sense since r controls the tree-width of the junction tree. On the other hand, q controls the number and length of branches in the junction tree. As q increases, for a fixed value of r, diagnosis time increases exponentially. Figure 4.17 which shows a plot of the log of diagnosis time against different values of q for $r = 2$ for diagnosis using junction trees and using Su and Wonham's method. In both cases, the diagnosis time increases exponentially with the value of q, but diagnostic performance of the JT algorithm is better by orders of magnitude.

In the case of S&W, the size of the largest structure quickly becomes unmanageable as q and r are increased. In contrast, for the JT method, the largest structure remains manageable and retains a fixed value dependent on r no matter what the value of q is for the cases considered.

4.5.3 Results on Ring Networks

Table 4.5 shows results obtained on ring networks with varying parameter values for n and k. As for branching networks, the size of the largest structure (synchronisation automaton) and the time for producing a diagnostic result were recorded for each network for two cases: one using Su and Wonham's (S&W) algorithm, and two using our proposed junction tree (JT) approach.

The results show that for small values of k ($k < 3$), the JT method shows better diagnostic time in general except when n is really large. However, when $k > 3$, S&W's algorithm produces

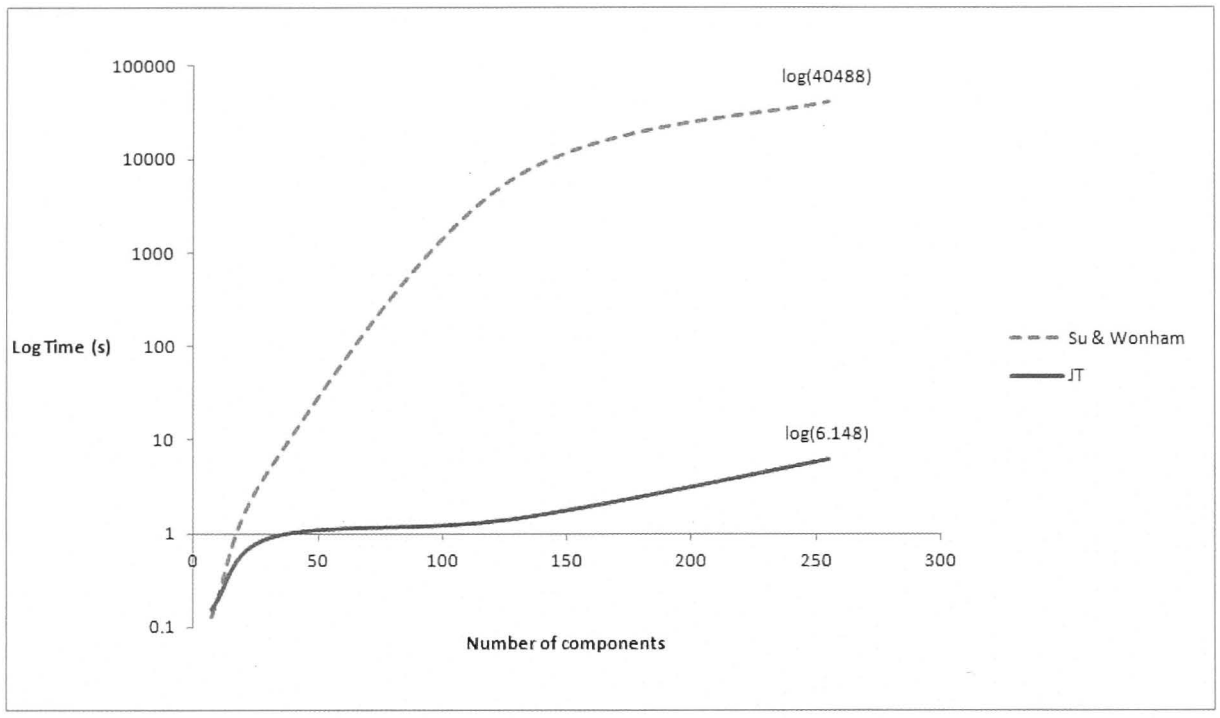


Figure 4.17: Graph of log of diagnosis time(s) versus number of components for a branch network configuration ($r = 2$)

n	k	Su & Wonham		JT	
		largest struc	time (s)	largest struc	time (s)
10	1	[4, 8]	0.264	[15, 50]	0.071
10	2	[36, 168]	0.893	[415, 2586]	0.916
10	3	[576, 4416]	11.136	[5714, 46954]	46.632
10	4	[6912, 69120]	497.752	[17148, 148896]	na
20	1	[4, 8]	0.459	[27, 108]	0.098
20	2	[4, 8]	3.447	[528, 3470]	1.664
20	3	[576, 4416]	598.857	na	na
30	1	[4, 8]	0.815	[27, 108]	0.324
30	2	[36, 168]	7.873	[528, 3470]	2.770
50	1	[4, 8]	0.815	[27, 108]	0.451
50	2	[36, 168]	20.004	[528, 3470]	6.360
100	1	[4, 8]	4.296	[27, 108]	0.749
100	2	[36, 168]	92.482	[528, 3470]	15.765
150	1	[4, 8]	5.452	[27, 108]	1.831
150	2	[36, 168]	203.493	[528, 3470]	24.210
200	1	[4, 8]	8.707	[27, 108]	4.035
2047	1	[4, 8]	2429.570	[27, 108]	4003.450

Table 4.5: Results on ring networks (where computation exhaust memory, the largest structures reached are shown in red)

faster diagnostic time. The largest structure obtained with the JT method has more states and events than the equivalent one obtained with the S&W method in every ring network we tested. In fact, in contrast to what is observed for branching networks, the size of the largest structure in the S&W method stays relatively small. Moreover, where the computation exhausts memory for the

JT algorithm, S&W is still able to produce a result without running out of memory despite that the computation time is large. We also note that the size of the largest structure shows a dependence on the value of k, and not of n, both for the JT method and S&W's method.

Figure 4.18 shows a graph of log of diagnosis time for different values of n when $k = 2$ for both the JT method and S&W's method. In both cases, the diagnosis time increases exponentially with the value of n although at a slower rate than for the branching networks in figure 4.17.

The results suggest that a combination of both methods, *e.g.* building a junction tree on the events of the system and using a heuristic ordering strategy for synchronisation, might further improve results on large networks that are highly connected.

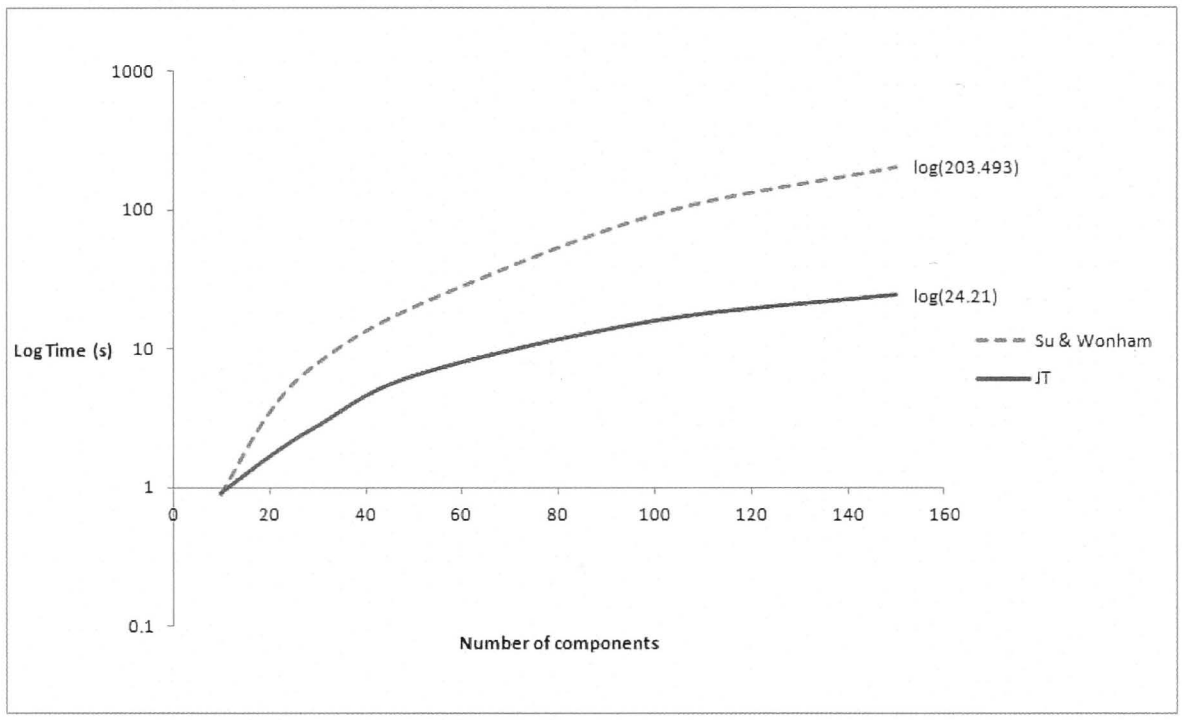


Figure 4.18: Graph of log of diagnosis time(s) versus number of components for a ring network configuration ($k = 2$)

4.5.4 Results on Random and Small-World Networks

Table 4.6 shows results obtained on small-world networks with varying parameter values for n_{ring}, k and p (the seed value used to generate p is kept constant with value 5 for all networks. Again, performance is measured for both the JT and S&W's methods.

The number of starting components connected into a ring network is given by n_{ring}. After applying the small-world transformation where connections are rewired with probability p, the actual number of components that are connected might be different. As it turned out for the cases presented in table 4.6, all the resulting small-world networks except one (the one with $n_{ring} = 200$) kept the same number of components as the starting ring networks they were built from. Random networks are special cases of small-world networks where $p = 1$.

Both algorithms performed very well on random and small-world networks with $k = 1$, although the JT method had faster diagnosis time. We note that S&W's algorithm took a very long time to run for the case where $(n_{ring} = 100, k = 1, p = 1)$ whereas the JT method completed in

n_{ring}	k	p	Su & Wonham		JT	
			largest struc	time (s)	largest struc	time (s)
10	1	0.1	[4, 8]	0.204	[24, 92]	0.043
10	1	0.5	[16, 64]	0.185	[12, 32]	0.029
10	1	1	[6, 14]	0.145	[7, 16]	0.047
20	1	0.1	[8, 24]	0.264	[26, 96]	0.052
20	1	1	[8, 24]	0.300	[217, 56]	0.097
50	1	0.1	[16, 64]	2.256	[30, 118]	0.297
50	1	0.2	[8, 24]	1.869	[29, 112]	0.313
50	1	0.3	[16, 64]	2.605	[33, 134]	0.204
50	1	0.4	[16, 64]	3.251	[25, 90]	0.326
50	1	0.5	[8, 24]	3.251	[25, 90]	0.326
50	1	0.6	[48, 256]	5.182	[43, 174]	0.223
50	1	0.7	[24, 104]	2.767	[32, 128]	0.264
50	1	0.8	[72, 408]	3.490	[21, 76]	0.205
50	1	0.9	[54, 270]	7.187	[17, 56]	0.239
50	1	1	[48, 256]	5.009	[13, 34]	0.204
100	1	0.1	[16, 64]	5.961	[36, 150]	1.061
100	1	0.2	[16, 64]	5.074	[24, 92]	0.554
100	1	1	[1024, 10240]	3295.955	[13, 36]	0.526
200[#]	1	0.1	[8, 24]	2.323	[21, 76]	0.254

[#] actual number of components after reconnection is 50

Table 4.6: Results on random and small-world networks (where computation exhaust memory, the largest structures reached are shown in red)

under a second. This is due to the fact that the structure of the network is very close to a tree structure and the diagnosis result obtained on it shows similarity with a case in a branching network ($q = 6, r = 2$) with roughly the same number of components (see table 4.4).

Figure 4.19 shows a graph of log of diagnosis time for small-world networks for the case where $k = 1$ and $p = 0.1$ using both the JT method and S&W's method. The diagnostic time for both methods was very good, even for networks with a large number of components. The graphs dip when $n_{ring} = 200$ because the number of components that are actually connected in the network is 50. The algorithm employed to generate small-world networks is stochastic. It so happened that every other network generated retained a number of connected components equal to n_{ring}.

Both methods do not cope well when $k > 1$; memory is exhausted before the algorithm terminates. Given that small-world networks can have characteristics of both branching networks (near-zero clustering coefficient) and of ring networks (high clustering coefficient), when they display a fairly equal balance of both characteristics, either diagnosis method struggles. This suggests an approach that applies S&W's method to parts of the network that have loops, and the JT method to parts that are tree-like. Investigating how to implement such an approach is material for future work.

Electricity networks (our motivating application) display small-world characteristics [98] but

with a very near-tree structure by virtue of their design. Hence use of the JT algorithm for diagnosis on such networks is justified. For cases where portions of the network display high clustering, we can adapt the method as suggested above. Another option is to selectively ignore some of the connections on the network when performing diagnosis without compromising accuracy of the results, a solution explored in the next chapter.

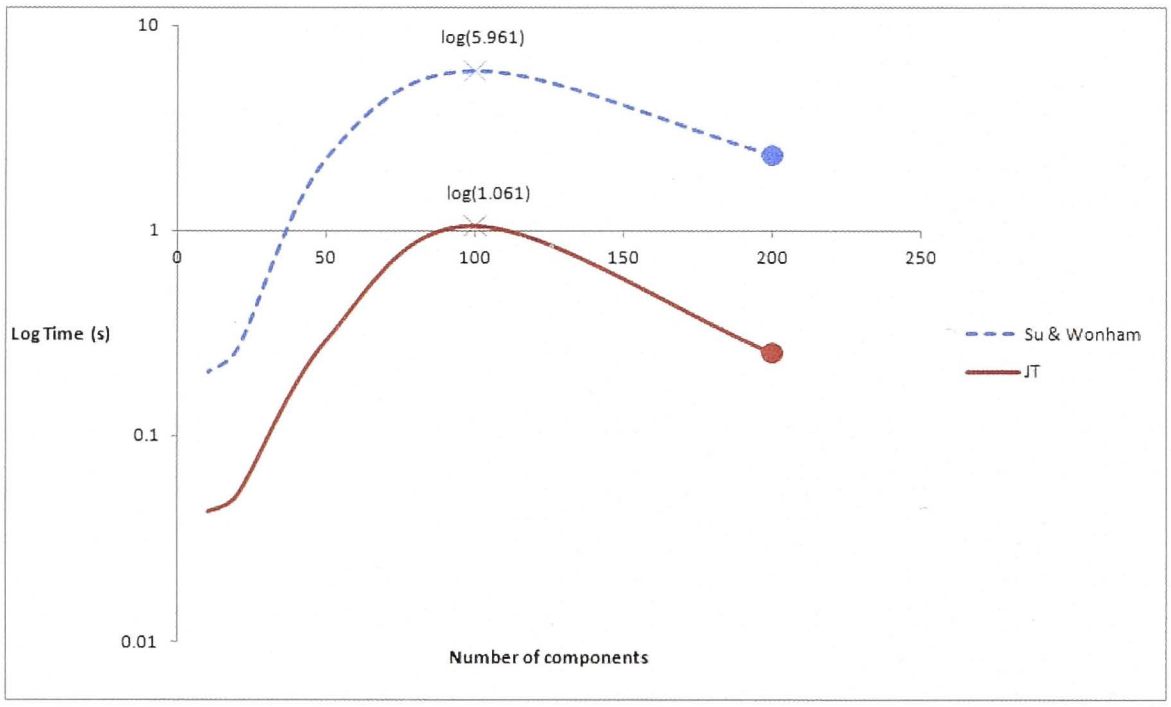


Figure 4.19: Graph of log of diagnosis time(s) versus number of components for a small-world network configuration ($p = 0.1$)

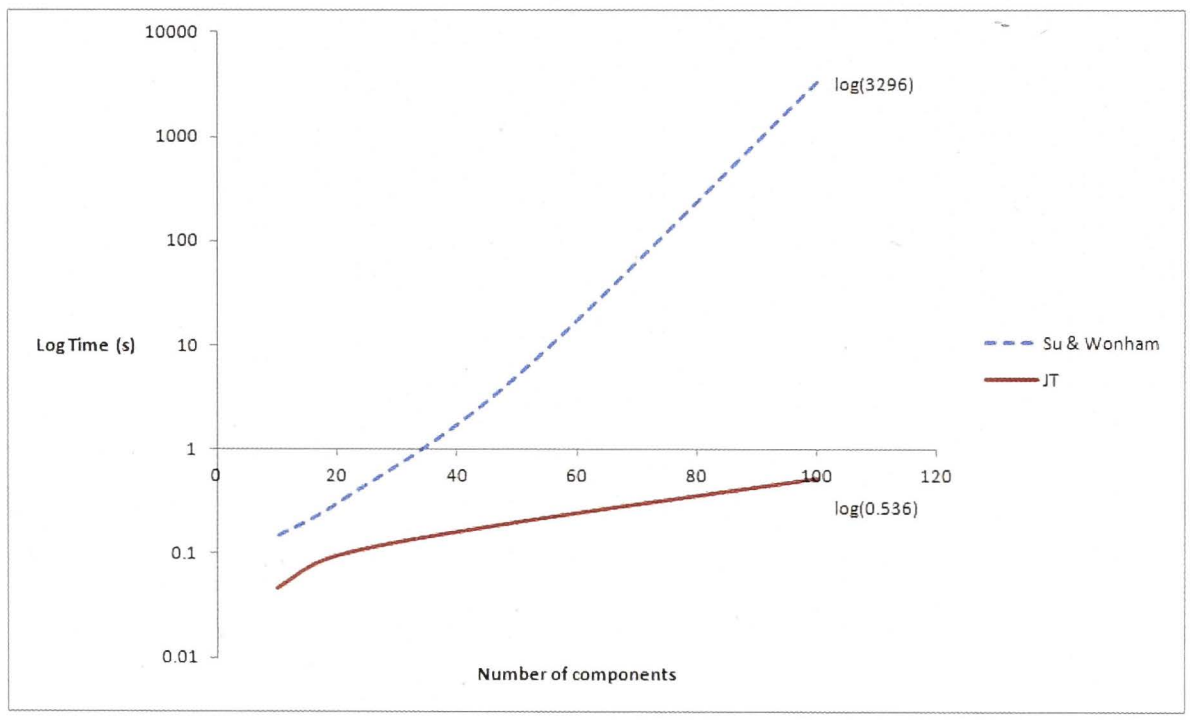


Figure 4.20: Graph of log of diagnosis time(s) versus number of components for a random network configuration (constructed using the small-world algorithm with $p = 1$)

4.6 Discussion

Using a junction tree is very interesting as the resulting subsystems tends to be of small size. However, this does not necessarily imply that the local diagnoses will actually be small as we show in the next example.

Consider a tree with n nodes N_1 to N_n. Each node N_i is associated with events e_{i-1} and e_i. The topology of the tree is thus simply a line as node N_i shares event e_i with node N_{i+1}. The automaton of each node N_i is represented in Figure 4.21. Since the initial state is the same as the final state, the number of occurrences of event e_i is twice that of event e_{i-1} for any i. Assume the node N_1 runs $k \in \mathcal{N}$ loops. Then, event e_0 occurs k times, event e_1 occurs $2 \times k$ times, etc. Event e_i occurs $2^i \times k$ times. The globally consistent automaton representing the behaviour on node N_i must represent the fact that event e_{i-1} occurred $2^{i-1} \times k$ times and the event e_i occurred $2^i \times k$ times for any natural number k (and not for rational non natural numbers). This requires $2^{i-1} + 2^i$ states and transitions. In this example, the number of states after local consistency is exponential in the number of nodes. ■

The result basically comes from the fact that the events e_i and e_j in this example are not concurrent events but they occur in sequence. We expect that most systems actually exhibit concurrent behaviours. In this case, the size of the local diagnosis on a cluster is a direct function of the number of events attached with this cluster, and thus smaller cluster lead to better efficiency.

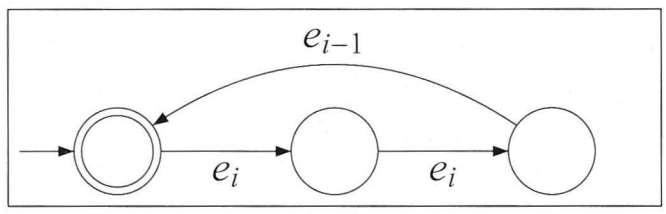


Figure 4.21: Automaton that models the language of node N_i

The natural topology of the system has an important impact on the quality of the produced junction tree, and hence the size of the subsystems. If we start off with a near tree-like structure, the resulting junction tree will produce smaller size clusters, and hence smaller automata to work with, reducing complexity. *E.g.* in Figure 4.4, graph 1 produces the best junction tree with smallest clusters (JT 1). With graph 3, because of the larger size clusters, the local diagnoses will actually be quite big (JT 3). The *tree-width* of a graph is the size of the largest cluster in its optimal junction tree minus one. Grids for instance have a tree-width linear in the size of the shortest side. Thus, we emphasize the importance of designing the system in a tree-like structure to make it easier to diagnose.

4.7 Conclusion

We identified the importance of a distribution of the system into (possibly overlapping) subsystems for the diagnosis of discrete-event systems. If the distribution generates a tree-shaped topology, an algorithm based on local consistency can ensure global consistency of the diagnosis. Simulation results on networks of different topologies confirm this. We used the graph theory of junction trees to obtain good distributions. The complexity of the diagnosis is then often bounded by the tree width of the system topology which places an upper bound on the number of automata to synchronise together, though counter-examples exist.

In this chapter, we proposed a static construction of the junction tree based only on the topology of the system. We want to investigate a more flexible technique where the junction tree is built after diagnoses and simple pruning operations are performed locally on components. The idea is that some connections in the system topology can be removed when no communication happened through these connections, leading to a graph with a smaller tree width. Moreover, we could then assign weight on each vertex of the graph. These technique should then improve the efficiency of diagnosis. More generally, we want to investigate more dynamic computations of junction trees: experiments have shown that the connections can often be removed after the distributed diagnosis is computed during the local consistency algorithm. For this reason, we want to start the diagnosis algorithm while the junction tree is being computed so as to dynamically change the construction of the junction tree. This is not trivial as the construction of the junction tree must satisfy relevant properties.

Regarding system design, an interesting exploration would be to interact with the system designer to propose alternative topology structures in the system in order to ensure a reasonable tree width of the system. We looked at a way to use sub-systems of a given system to do diagnosis. The complexity of the topology of a subsystem is lower than that of the whole system and thus the tree-widths that we need to handle are smaller. We present this extension in Chapter 5.

Finally, we considered that the observations emitted by different components were completely independent. However, it is often the case that a (partial) order exists between the observations. *E.g.* the alarm emitted by component 1 was surely emitted before the alarm from component 2. This generates some kind of connection between the two components and potentially interconnect all the components. We want to investigate this issue and determine when these connections can be removed, possibly with an approach based on time slicing [29].

Chapter 5

Augmented Distributed Diagnosis with Accuracy Criterion

5.1 Motivation

Using decentralised techniques helps in limiting combinatorial explosion when diagnosing large discrete event systems, but is not sufficient. Often, the complexity of the diagnosis is dependent on how components in the system are connected and the number of connections between them.

We propose in this chapter to augment the decentralised junction tree-based approach presented in Chapter 4 by ignoring some connections on the system. This helps reduce the complexity, and hence the cost, of the diagnostic reasoning required. However, accuracy of the diagnosis is also reduced. We get around this problem by performing an off-line analysis to determine which connections can be safely ignored.

Ignoring certain connections makes it possible to reason on smaller subsystems such that diagnosis can be obtained in reasonable time. However, this could lead to a loss of accuracy of the diagnosis. This results from not taking into account information from the ignored connections that could have helped in eliminating certain diagnostic scenarios. Therefore, we perform a prior *accuracy analysis* on the model to determine which connections can be ignored without having a negative impact on the global accuracy of the diagnoser.

5.2 Preliminaries with augmented notation

We use the same language formalism as in Chapter 4. However, the notation is augmented in order to handle the subtleties introduced in this chapter and explain the concepts introduced in this chapter. In particular, we want to be able to handle events that appear simultaneously.

We denote by Σ a set of *symbols* (modeling events on the system). A word σ is a finite sequence of sets of symbols $s_1.\cdots.s_n$ such that $\forall i,\ s_i \subseteq \Sigma, s_i \neq \emptyset$. So, if $\Sigma = \{a, b, c, d\}$, then $\{a\}.\{b, c\}.\{d\}.\{a\}$ is a word on Σ where b and c appear simultaneously.

The empty sequence is denoted ε. We abuse notation and write $s_i \in \sigma$ to denote that s_i is in the sequence σ. $(2^{\Sigma})^{\star}$ is the set of words on Σ. A *language* $\mathcal{L}$ on Σ is a subset of words $\mathcal{L} \subseteq (2^{\Sigma})^{\star}$. The *projection* operation can be used to focus on specific events of $\Sigma' \subseteq \Sigma$.

Generally, a word is defined simply as a sequence of symbols; we use an augmented notation so that we can represent the occurrence of simultaneous events and gain more flexibility to describe the main contribution of this chapter that is about the relaxation of connections. Events can be shared between components and can thus happen at the same time. However when a connection is relaxed, it is then possible that these connected events occur separately. Hence, the augmented notation is necessary to distinguish them.

Definition 36 (Augmented Projection)**.** The projection on Σ' of a word σ on $\Sigma \supseteq \Sigma'$, denoted $\mathrm{P}_{\Sigma\to\Sigma'}(\sigma)$, or simply $\mathrm{P}_{\Sigma'}(\sigma)$, is the word on Σ' that only retains the symbols of Σ' and removes empty symbol sets. Formally,

$\mathrm{P}_{\Sigma\to\Sigma'}(\sigma) =$

$$\begin{cases} \varepsilon & \text{if } \sigma = \varepsilon, \\ \mathrm{P}_{\Sigma\to\Sigma'}(\sigma') & \text{if } (\sigma = s.\sigma') \wedge (s \cap \Sigma' = \emptyset), \\ s.\,\mathrm{P}_{\Sigma\to\Sigma'}(\sigma') & \text{if } (\sigma = s.\sigma') \wedge (s \cap \Sigma' \neq \emptyset). \end{cases}$$

The projection on Σ' of the language $\mathcal{L}$ on $\Sigma \supset \Sigma'$, denoted $\mathrm{P}_{\Sigma\to\Sigma'}(\mathcal{L})$, is the set of words in $\mathcal{L}$ projected onto Σ': $\mathrm{P}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\mathrm{P}_{\Sigma\to\Sigma'}(\sigma) \mid \sigma \in \mathcal{L}\}$. The inverse operation gives the set of words on Σ whose projection on Σ' is included in the language of origin: $\mathrm{P}^{-1}_{\Sigma\to\Sigma'}(\mathcal{L}) = \{\sigma \in (2^{\Sigma})^{\star} \mid \mathrm{P}_{\Sigma\to\Sigma'}(\sigma) \in \mathcal{L}\}$.

Synchronisation

Each local entity has its own specific language to represent its behaviour. When several entities are concerned, we need to *synchronise* their languages to generate a globally consistent language. Each language has its own symbol set, disjoint from the symbol sets of other languages. However, some symbols from different local sets could be different representations of the same physical reality. The synchronisation operation coordinates these equivalent symbols by forcing their simultaneity. Equivalent symbols on different languages are represented by synchronous sets.

Definition 37 (Synchronous Set)**.** Given two disjoint sets of symbols Σ_1 and Σ_2, a synchronous set $\mathcal{S}$ is a set of symbol pairs coming from the two sets: $\mathcal{S} \subseteq \Sigma_1 \times \Sigma_2$.

An element $\langle a, b\rangle \in \mathcal{S}$ indicates that a and b are describing the same physical reality and we have to ensure that they are considered simultaneously.

Definition 38 (Augmented Language Synchronisation)**.** Given two languages $\mathcal{L}_1$ on Σ_1 and $\mathcal{L}_2$ on Σ_2, and a synchronous set $\mathcal{S}$ defined on Σ_1 and Σ_2. The synchronous product of $\mathcal{L}_1$ and $\mathcal{L}_2$

on $\mathcal{S}$, denoted $\mathcal{L}_1 \bigotimes_{\mathcal{S}} \mathcal{L}_2$,is defined as the set of words on $(\Sigma_1 \cup \Sigma_2)$ whose projection on each local symbol set is the local language, and satisfies the constraint of simultaneity introduced by $\mathcal{S}$. Formally: $\{\sigma \in (2^{\Sigma_1 \cup \Sigma_2})^\star \mid (\forall i \in \{1,2\},\ \mathrm{P}_{\Sigma_1 \cup \Sigma_2 \to \Sigma_i}(\sigma) \in \mathcal{L}_i) \wedge (\forall \langle a, b\rangle \in \mathcal{S},\ \forall s \in \sigma,\ a \in s \Leftrightarrow b \in s)\}$.

It is possible to prove that these notations preserve the properties of commutativity and associativity of the more traditional notations (although this would imply redefining synchronous sets). For simplicity and where it is obvious, the set $\mathcal{S}$ can be dropped from the notation: $\mathcal{L}_1 \otimes \mathcal{L}_2$.

We introduce one last augmented notion here: *local consistency*.

Definition 39 (Augmented Local Consistency)**.** The local consistency operation between two languages $\mathcal{L}_1$ and $\mathcal{L}_2$ builds the smallest language $\mathcal{L}_1'$ that maintains $\mathcal{L}_1' \otimes \mathcal{L}_2 = \mathcal{L}_1 \otimes \mathcal{L}_2$. This operation can be implemented by: $\mathcal{L}_1' = \mathrm{P}_{\Sigma_1}(\mathcal{L}_1 \otimes \mathcal{L}_2)$.

5.2.1 Example Representation of a Distributed System

We illustrate the use of language representation using automata on a distributed network consisting of four components A, B, C and D as represented in figure 5.1.

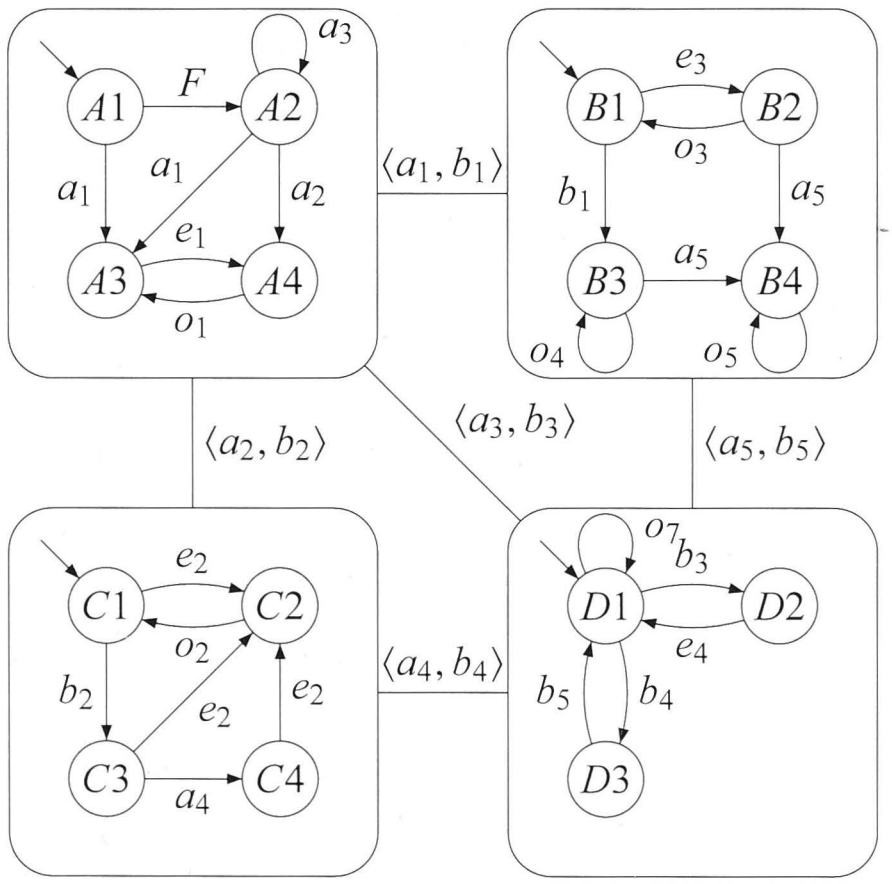


Figure 5.1: Example of network

Figure 5.1 gives an example of language represented in a distributed fashion. The language $\mathcal{L}$ is defined by four languages $\mathcal{L}_A$ to $\mathcal{L}_D$ that are synchronised through five set of simultaneous events $\langle a_1, b_1\rangle$ to $\langle a_5, b_5\rangle$; each local language is represented by an automaton. The synchronisation of languages $\mathcal{L}_B$ and $\mathcal{L}_D$ is represented on Figure 5.2.

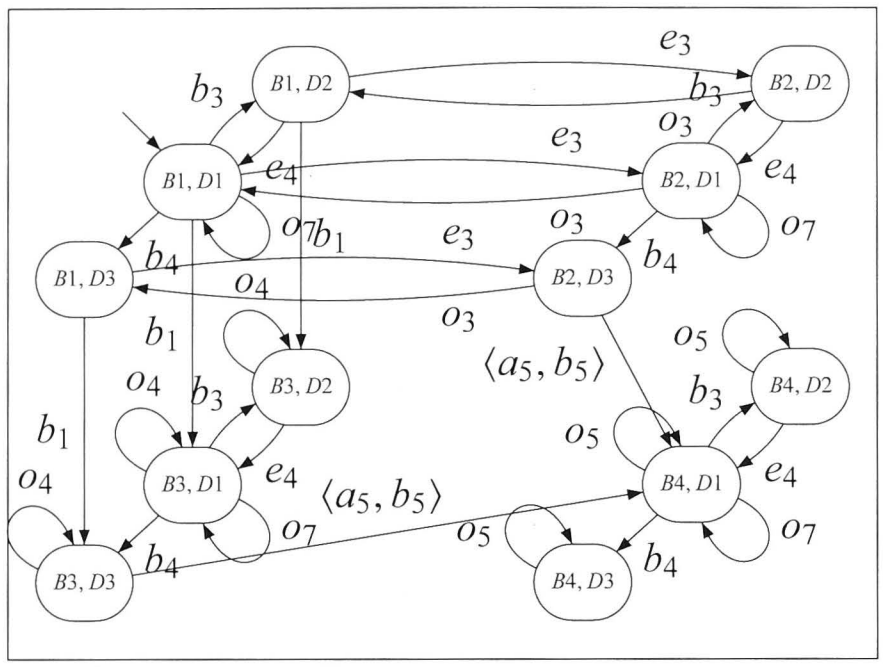


Figure 5.2: Synchronisation of $\mathcal{L}_B$ and $\mathcal{L}_D$

5.3 Diagnoser, Explanatory Language and Connections

We clarify, in this section, a few concepts that are needed to explain the contribution of this chapter. Some have been introduced previously, but require redefining in the modified augmented notation adopted for this chapter.

5.3.1 Model

We assume we have a complete model of a system Γ captured by a language *Mod* on a finite set of events Σ (*i.e.* $Mod \subseteq (2^{\Sigma})^{\star}$). The set of faulty events is denoted $\Sigma_F \subseteq \Sigma$.

Some events generate the emission of an observation. These *observable* events are denoted $\Sigma_o \subseteq \Sigma$. The observations are represented by a language $Obs \subseteq (2^{\Sigma_o})^{\star}$. We assume there is no noise on the observations, so that *Obs* contains only one element that is a sequence of observable events (see [29]).

5.3.2 Diagnoser

A diagnoser is an agent in charge of monitoring the observations generated by a system to provide diagnosis reports. A diagnoser dedicated to a specific fault $F \in \Sigma_F$, denoted Δ^F, may return one of the three following results:

1. F-sure: the fault occurrence is asserted;
2. F-safe: the fault occurrence is disproved;
3. F-ambiguous: the fault occurrence is unknown.

By definition, the fault F is considered permanent. For each fault $F \in \Sigma_F$, we define an agent Δ^F responsible for the detection of F. The rest of the article focusses on a single fault event F and we

therefore simplify the notation Δ^F to Δ.

5.3.3 Explanatory Language

The explanatory language is the set of behaviours accepted by the model and consistent with the observations (as defined previously in section 2.2.6.1). It can be defined as follows:

$$Expl = Mod \otimes Obs\,.$$

From *Expl*, it is possible to compute the global F-diagnoser Δ_{Mod} associated with F: that is, for any Obs, $\Delta_{Mod}(Obs) =$

$$\begin{cases} \text{F-sure} & \text{if } \forall \sigma \in Expl, \exists s \in \sigma : F \in s \\ \text{F-safe} & \text{if } \forall \sigma \in Expl, \exists s \in \sigma : F \notin s \\ \text{F-ambiguous} & \text{otherwise.} \end{cases}$$

Whichever representation is chosen for languages (automaton, Petri nets, etc.), diagnosis faces the problem of search-space explosion. The reasoning is exponentially complex with the number of components in the system, which makes trivial techniques impossible to apply for systems with a few dozens components. Distributed techniques aim at tackling this issue.

5.3.4 Connection

We describe again a distributed system before defining what we mean in context by a *connection*. Modern technical systems usually consist of components that are each an individual system with simple behaviours, but interacts with other components to produce an overall complex behaviour. We refer to the overall system as a distributed system and model each of its component separately. Let Γ be a distributed system made up of a set of components: $\Gamma = \{\Gamma_1 \ldots \Gamma_n\}$. Each component Γ_i can be described by the language Mod_i defined on the alphabet Σ_i. The implicit assumption of fairness is made, whereby components cannot become silent in the long run: on an infinite time-scale, the number of observations generated by a given component is always infinite. Fault events are intrinsic to a component's physical set-up which is responsible for causing failures on the component itself but also causing them to propagate over the system. The occurrence of a fault of type F is considered as an event that can only happen on a component $\Gamma_i : F \in \Sigma_i \wedge (i \neq j \Rightarrow F \notin \Sigma_j)$. Components in a system communicate through *connections*.

Definition 40. (Connection) A connection $\mathcal{K}_{ij}$ exists between two components Γ_i and Γ_j if they have a physical or logical link between them that allows the exchange of information about the events occurring in each of them. A synchronous set $\mathcal{S}_{ij}$ can be used as abstract model for a connection $\mathcal{K}_{ij}$ where $\mathcal{S}_{ij} \subseteq \Sigma_i \times \Sigma_j$ and $\mathcal{S}_{ij} = \mathcal{S}_{ji}$. $\langle b, a \rangle \in \mathcal{S}_{ji} \Rightarrow \langle a, b \rangle \in \mathcal{S}_{ij}$

We make the assumption that an event can only be part of one connection. $\forall i, j, \Sigma_i \cap \Sigma_j \neq \emptyset \Rightarrow i = j$.

The way in which the components of a distributed system are connected has direct impact on the global *topology* of the system. The global model of the system is implicitly defined by synchronising the models for all components of the system ($Mod = Mod_1 \otimes \cdots \otimes Mod_n$), hence it is unnecessary to calculate it explicitly. Observations on the system can also be modeled in a decentralised fashion: $Obs = Obs_1 \otimes \cdots \otimes Obs_n$ [29].

The explanatory language $Expl_i$ on a component Γ_i is given by $Mod_i \otimes Obs_i$. The global explanatory language $Expl$ is calculated by obtaining the synchronous product of the local languages:

$$\begin{aligned} Expl &= (Mod_1 \otimes \ldots \otimes Mod_n) \otimes (Obs_1 \otimes \ldots \otimes Obs_n) \\ &= (Mod_1 \otimes Obs_1) \otimes \ldots \otimes (Mod_n \otimes Obs_n) \\ &= Expl_1 \otimes \ldots \otimes Expl_n. \end{aligned}$$

Calculating the language $Expl$ by synchronising all components is often impossible if the system consists of a large number of components. *Distributed* methods of diagnosis helps avoiding this calculation. We use a junction-tree based implementation as in chapter 4.

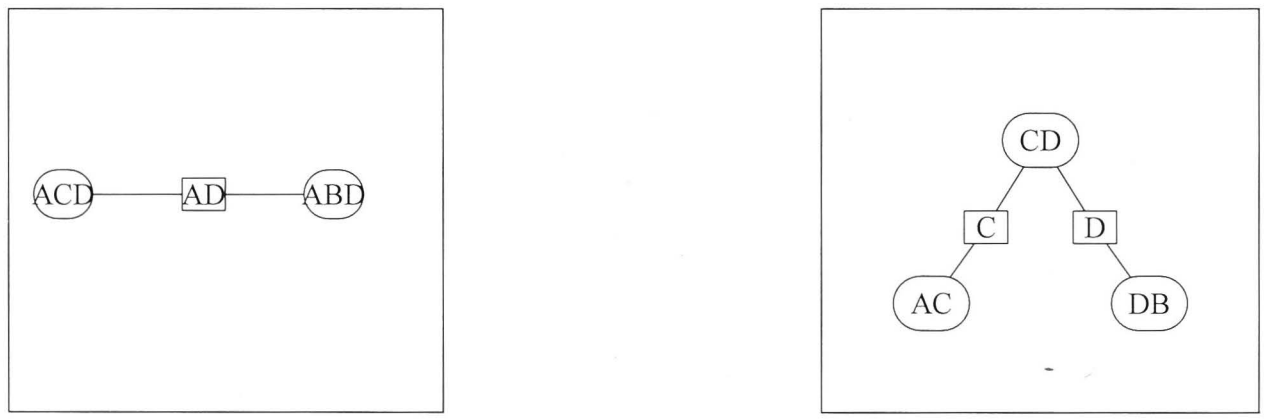


Figure 5.3: Junction Tree for whole system in Figure 5.1 (left) and same system with connections $\langle a_1, b_1 \rangle$, $\langle a_2, b_2 \rangle$ removed (right)

Consider a graph $\mathcal{G} = \langle \Gamma, \mathcal{K} \rangle$ on the components of system Γ where $\mathcal{K}$ is the set of all connections on the system. A junction tree [49] on $\mathcal{G}$ is a pair $(\mathcal{J}, C)$ where $\mathcal{J}$ is a tree and C is a function that associates each node $\mathcal{N}$ of $\mathcal{J}$ to a *cluster* of components C_i (see example figure 5.3). Moreover, for each connection $\langle i, j \rangle$, there exists a cluster containing the nodes : $\{i, j\} \subseteq C(\mathcal{N})$. Finally, if two clusters of the tree contains the same node, every cluster between them will contain that node (see the node D between the clusters ABD and DEF).

To obtain the diagnosis on a distributed system, it is sufficient to calculate the local explanatory language of each cluster and to perform local consistency operations on the junction tree from the leaves to the root and back from the root to the leaves. If the root of the tree is chosen to contain the component on which an event occurs, the global diagnosis of the fault is obtained using formula (5.3.3) on the local explanatory language of the root.

This method allows us to circumvent the explicit calculation of $Expl$. However, calculating the explanatory language on each cluster is still necessary for each cluster, and the complexity of the representation of this language increases exponentially with the number of components in

the cluster. The *tree-width* of the topology is the size of the biggest cluster of its junction tree minus one. This value serves as an apriori estimate of the algorithmic cost of diagnosis by this method. Hence, if we limit the tree-width we are potentially able to reduce the cost of diagnosis. We present in the next section a proposed method to handle this.

5.4 Diagnosis with sub-configuration and Accuracy

The effectiveness of the diagnosis algorithm using junction trees is directly dependent on the connections between components in the system. We therefore propose to relax some of these connections with the goal of generating a tree on which reasoning can be carried out more effectively.

5.4.1 Relaxation of connections

The relaxation of connections is formalised with sub-topology and sub-configuration notions.

Definition 41 (Sub-topology)**.** A *sub-topology* $\mathbb{T}$ on a distributed system Γ is a subset of connections $\mathcal{Y} \subseteq \mathcal{K}$.

A sub-topology $\mathbb{T}$ defines a language $\mathcal{L}(\mathbb{T})$ that corresponds to the synchronisation of local languages on the connections of the set $\mathcal{Y}$. We illustrate this using the example in Figure 5.1. This system consists of four components A to D and five connections $\langle a_i, b_i \rangle$. A possible word on the system is $\{F\}.\{e_3\}.\{a_2, b_2\}.\{a_4, b_4\}.\{a_5, b_5\}.\{o_5\}$ (we note that each a_i is synchronised with a corresponding b_i). We now consider the sub-topology where the connection $\langle a_2, b_2 \rangle$ is ignored. The language of this sub-topology contains additional words, including $\{e_3\}.\{b_2\}.\{a_4, b_4\}.\{a_5, b_5\}.\{o_5\}$ (here b_2 appears on its own).

Lemma 1. Words defined on a sub-topology $\mathbb{T} \subseteq \mathbb{T}'$ need to satisfy less constraints than those of $\mathbb{T}'$: $\mathcal{L}(\mathbb{T}') \subseteq \mathcal{L}(\mathbb{T})$, where $\mathcal{L}$ represents either the system model or the explanatory language.

In practice, a sub-topology can isolate components of the system. In that case, it becomes unnecessary to keep track of the observations of those components since the model indicates that they are functioning independently from the other components. This is encompassed by the notion of a sub-configuration.

Definition 42 (Sub-configuration)**.** A *sub-configuration* $\mathbb{C}$ is a tuple $(\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_\mathbb{C}, \overline{\mathcal{Y}_\mathbb{C}})$ where $\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}$ is a set of components, $\mathcal{Y}_\mathbb{C}$ is a set of connections between the components of $\mathbb{C}$, and $\overline{\mathcal{Y}_\mathbb{C}}$ is the set of connections between the components of $\mathbb{C}$ that are not found in $\mathcal{Y}_\mathbb{C}$.

Figure 5.4 illustrates two different sub-topologies from the example of Figure 5.1 (on the left hand side the sub topology is $\{\langle a_2, b_2 \rangle, \langle a_4, b_4 \rangle, \langle a_5, b_5 \rangle\}$ and on the right hand side the sub-topology is $\{\langle a_2, b_2 \rangle\}$). The corresponding sub-configuration on the left hand side (resp. on the right hand side) involves the components $\{A, B, C, D\}$ (resp. $\{A, C\}$).

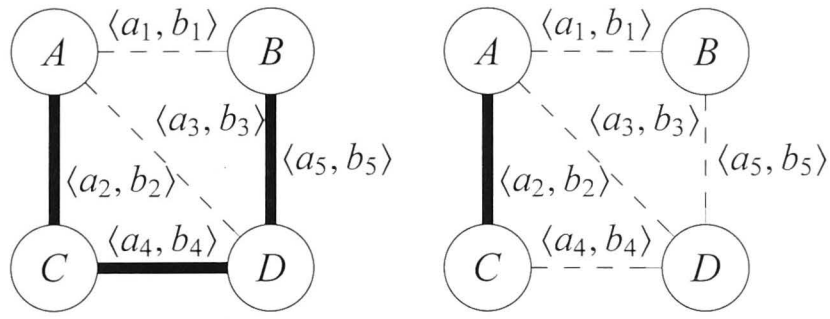


Figure 5.4: Two different sub-topologies - solid lines represent connections under consideration and dotted lines represent connections that are ignored

		Δ_{Mod}		
		F-sure	F-safe	F-amb
$\Delta_{Mod'}$	F-sure	✓	×	×
	F-safe	×	✓	×
	F-amb	A	A	✓

Table 5.1: Comparison between diagnosis results

5.4.2 Diagnosis within a sub-configuration

The basic idea of the work in this chapter is to perform diagnosis based on a model (denoted Mod' here) that is simpler than the model Mod. Table 5.1 represents what can be expected by doing so. Each cell indicates the diagnosis result of model-based diagnosis using the original model Mod compared to the simplified model Mod'. The diagonal (labels ✓) represents the cases where $\Delta_{Mod'}$ returns the same result as the original diagnoser Δ_{Mod}. The cell labeled A shows an accuracy reduction: diagnoser Δ_{Mod} can decide whether a fault occurred while $\Delta_{Mod'}$ cannot. The label × indicates inconsistent cases: the simplified model Mod' is inconsistent with the model Mod which means the diagnoser $\Delta_{Mod'}$ returns inconsistent results with regards to Δ_{Mod}. Regarding this table, it is better to determine simplified models Mod' such that diagnostic results correspond to cells labeled by ✓. Cells labeled by A are acceptable in the sense that they only betray loss of accuracy. However, the model Mod' should be chosen such that the cells labelled × are unreachable.

It is easy to demonstrate that model-based diagnosis using, as a simplified model, a sub-topology $\mathbb{T}$ (or its equivalent sub-configuration $\mathbb{C}$) falls in the acceptable category. Indeed, as stated in 1, the generated language by $\mathbb{T}$ always contains the initial language, so the corresponding diagnoser cannot provide inconsistent results but in the worst case less accurate results.

5.4.3 Accurate diagnoser on $\mathbb{C}$

We define a diagnoser $\Delta_{\mathbb{C}}$ on a sub-configuration $\mathbb{C}$ to be the diagnosis result obtained by using $\mathbb{C}$ as simplified model Mod'. $\Delta_{\mathbb{C}}$ is obtained by synchronising the language defined on $\mathbb{C}$, $\mathcal{L}(\mathbb{C})$, with observation Obs on the system: $\Delta_{\mathbb{C}} = \mathcal{L}(\mathbb{C}) \otimes Obs$.
The challenge is now to determine a sub-configuration $\mathbb{C}$ based on which the diagnoser $\Delta_{\mathbb{C}}$ maintains the accuracy with respect to the global diagnoser Δ_{Mod}. Formally,

Definition 43 (Accuracy)**.** The diagnoser $\Delta_{\mathbb{C}}$ is said to be *accurate* if for every observable σ_o emitted from the system such that $\Delta_{Mod}(\sigma_o)$ = F-sure, and for every continuing observable σ'_o of the system, there exists a bound $n \in \mathbb{N}$ such that $|\sigma'_o| \geq n$, $\Delta_{\mathbb{C}}(\mathrm{P}_{\mathcal{L}(\mathbb{C})}(\sigma_o.\sigma'_o))$ = F-sure (see Figure 5.5).

Figure 5.5 illustrates the concept of Accuracy. It shows the relative point in time at which a global diagnoser Δ_{Mod} becomes accurate (top row) and the point at which a diagnoser on sub-configuration $\Delta_{\mathbb{C}}$ becomes accurate (bottom row). The difference between those two points is given by the bound n.

Diagnoser accuracy is possible only under the assumption of observability fairness in the system (see section 4).

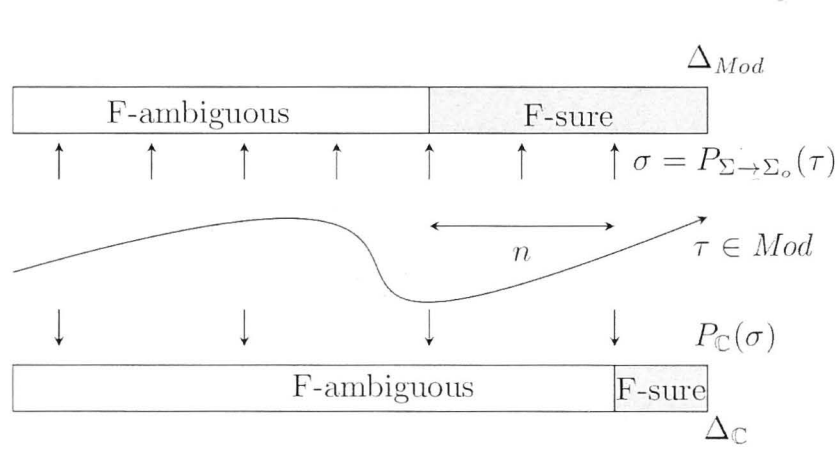


Figure 5.5: Accurate diagnoser $\Delta_{\mathbb{C}}$.

The main attraction of an accurate diagnoser $\Delta_{\mathbb{C}}$ is its ability to eventually obtain the same result, albeit with a finite delay, as a global diagnoser Δ_{Mod} if the fault F has occurred. In fact, as soon as a fault F has occurred on the system, Δ_{Mod} has two possible answers to explain the current sequence σ_o of observations: either it responds F-ambiguous or F-sure. By the fairness property of the system, $\Delta_{\mathbb{C}}$ also responds as soon as a new observation o is available on $\mathbb{C}$. Let $\sigma''_o.o$ be this finite continuation of σ_o, if $\Delta_{Mod}(\sigma)$ = F-ambiguous, there are two possible scenarios:

1. either the ambiguity is still present $\Delta_{Mod}(\sigma_o\sigma''_o.o)$ = F-ambiguous, then by construction, $\Delta_{\mathbb{C}}(\mathrm{P}_{\mathbb{C}}(\sigma_o\sigma''_o.o))$ = F-ambiguous = $\Delta_{Mod}(\sigma_o\sigma''_o.o)$;
2. or the ambiguity is no longer present $\Delta_{Mod}(\sigma_o\sigma''_o.o)$ = F-sure and then, by waiting a finite number n of observations σ'_o, $\Delta_{\mathbb{C}}(\mathrm{P}_{\mathbb{C}}(\sigma_o\sigma''_o.o\sigma'_o))$ = F-sure, and therefore in the end, $\Delta_{\mathbb{C}}$ returns the same result as Δ_{Mod} but by only observing $\mathbb{C}$.

5.4.4 Characterisation of an accurate diagnoser

In order to determine whether the diagnoser $\Delta_{\mathbb{C}}$ is accurate or not for a given sub-configuration $\mathbb{C}$, it is thus sufficient to analyse apriori if the sub-configuration $\mathbb{C}$ contains the characteristics that are required to implement an accurate diagnoser on it. Before describing these characteristics, some notations are introduced.

We consider a sub-configuration $\mathbb{C} = \{\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_{\mathbb{C}}, \overline{\mathcal{Y}_{\mathbb{C}}}\}$. We assume that the fault F has to occur on one of the components $\Gamma_F = \Gamma_{p_i}$ of $\mathbb{C}$. We also introduce the sub-configuration $\mathbb{C}_{\max} = \{\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}, \mathcal{Y}_{\mathbb{C}} \cup \overline{\mathcal{Y}_{\mathbb{C}}}, \emptyset\}$ that is associated with $\mathbb{C}$ in which no connection is relaxed. $\mathbb{C}_{\max}$ therefore takes into consideration all connections of the system that involve the components $\{\Gamma_{p_1}, \ldots, \Gamma_{p_m}\}$. The language defining the events generated by the sub-configuration $\mathbb{C}$ (resp. $\mathbb{C}_{\max}$) is denoted $\mathcal{L}_{\mathbb{C}}$ (resp. $\mathcal{L}_{\mathbb{C}_{\max}}$). By definition, $\mathcal{L}_{\mathbb{C}_{\max}} \subseteq \mathcal{L}_{\mathbb{C}}$. In this section, to simplify, Σ is constrained to the set of events of $\mathbb{C}$ (and therefore of $\mathbb{C}_{\max}$). Among the events of Σ we distinguish in particular: the set Σ_o of observable events, the set Σ_r^{ext} of interactive events of $\mathbb{C}$ associated with external relaxed connections (*i.e.* a connection of the system where only one of the components belong to $\mathbb{C}$). Finally, as it will be explained later on, the characterisation of an accurate diagnoser relies on the notion of *traces* and *observable traces*.

Definition 44 (Trace)**.** Let $F \in \Sigma$ be a fault and $\mathbb{C}$ a sub-configuration, the set of *traces* of F in $\mathbb{C}$ is the language :

$$\mathcal{T}(\mathbb{C}, F) = \{\tau = s_1 \ldots . s_m \in \mathcal{L}_{\mathbb{C}}, \exists s_i, F \in s_i\}$$

with $s_i \subseteq \Sigma, i \in \{1, \cdots, m\}$.

Similarly, the complement of $\mathcal{T}(\mathbb{C}, F)$ in $\mathcal{L}_{\mathbb{C}}$ (denoted $\mathcal{T}(\mathbb{C}, \neg F)$) consists of the set of traces where the fault F is not present. Figure 5.6 illustrates the traces associated to fault F in the sub-configuration consisting only of the component A of Figure 5.1.

Definition 45 (Observable Trace)**.** Let $F \in \Sigma$ be a fault and $\mathbb{C}$ a sub-configuration, an *observable trace* of F in $\mathbb{C}$ is a sequence of observable events of the language :

$$Obs(\mathbb{C}, F) = \mathrm{P}_{\Sigma \to \Sigma_o}(\mathcal{T}(\mathbb{C}, F)).$$

Similarly, $Obs(\mathbb{C}, \neg F)$ represents the set of observable traces of $\mathbb{C}$ where F is not present (see Figure 5.6).

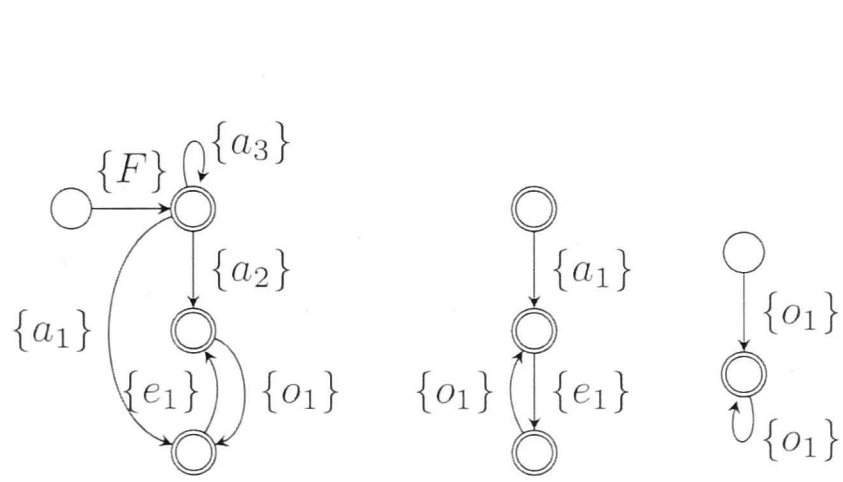


Figure 5.6: Traces $\mathcal{T}(A, F)$, $\mathcal{T}(A, \neg F)$ and $Obs(A, F)$.

We first explore the reasons why the diagnoser $\Delta_{\mathbb{C}}$ is not accurate for a given sub-configuration $\mathbb{C}$. We then describe the necessary criteria for making $\Delta_{\mathbb{C}}$ accurate.

5.4.4.1 What are the sources of inaccuracy in $\mathbb{C}$?

Let $\sigma_o.o$ be an observable sequence of the system which ends in the observable event o from $\mathbb{C}$ and for which the global diagnoser returns $\Delta_{Mod}(\sigma_o.o)$ = F-sure. Let $\sigma'_o.o$ be the observable projection of $\sigma_o.o$ on $\mathbb{C}$. Firstly, the answer $\Delta_{\mathbb{C}}$ to the observation $\sigma'_o.o$ can only be F-sure or F-ambiguous as $\sigma'_o.o$ has to be an observable trace of F in $\mathbb{C}$. Secondly, if $\Delta_{\mathbb{C}}(\sigma'_o.o)$ = F-sure, there is no accuracy problem. There only remains the problematic case of $\Delta_{\mathbb{C}}(\sigma'_o.o)$ = F-ambiguous while $\Delta_{Mod}(\sigma_o.o)$ = F-sure. In this case, $\sigma'_o.o$ is an ambiguous observable trace ($\sigma'_o.o \in Obs(\mathbb{C}, F) \cap Obs(\mathbb{C}, \neg F)$) and this ambiguity is always due to the following situations.

1. The set of components of $\mathbb{C}$ are not sufficiently observable locally and only observations emitted from components external to $\mathbb{C}$ can eliminate the ambiguity (this problem is intrinsic to $\mathbb{C}_{\max}$).
2. There are too many relaxed connections in $\mathbb{C}$. The diagnosis is ambiguous because $\Delta_{\mathbb{C}}$ assumes the existence of behaviours that are not possible in $\mathbb{C}_{\max}$.

The difficulty now lies in determining a criterion on the configuration $\mathbb{C}$ that guarantees that, if the diagnosis of $\Delta_{\mathbb{C}}$ is ambiguous then that of Δ_{Mod} also is. That criterion must guarantee that none of the two situations above hold in the sub-configuration $\mathbb{C}$. As the first situation is intrinsic to $\mathbb{C}_{\max}$ and the second is due to relaxation of connections, we first determine such a criterion on $\mathbb{C}_{\max}$ sub-configurations only.

5.4.4.2 Detection criterion of the accuracy of $\mathbb{C}_{\max}$

The sequence $\sigma_o.o$, introduced above, represents an observable sequence of Γ and therefore there exists at least one trace τ of Γ such that $Obs(\tau) = \sigma_o.o$. Let $\tau_{int} = P_{\Sigma_r^{ext}}(\tau)$ be the interactive trace issued from τ and associated to the configuration $\mathbb{C}_{\max}$, then the detection criterion depends on the following results.

Property 1. If there exist in $\mathbb{C}_{\max}$ two traces τ_F and $\tau_{\neg F}$ such that :

- $P_{\Sigma_r^{ext}}(\tau_F) = P_{\Sigma_r^{ext}}(\tau_{\neg F}) = \tau_{int}$
- $P_{\Sigma_o}(\tau_F) = P_{\Sigma_o}(\tau_{\neg F}) = \sigma'_o.o$
- $F \in \tau_F \wedge F \notin \tau_{\neg F}$

then $\Delta_{Mod}(\sigma_o.o)$ = F-ambiguous.

Proof : the result is immediate. Considering that τ_F forms part of a global trace that can explain $\sigma_o.o$, the trace $\tau_{\neg F}$ necessarily forms part of another global trace $\sigma_o.o$ (since they have the same observable and interactive projections). Finally, there indeed exist two global traces that explain $\sigma_o.o$, one containing F and one not. □

Property 1 describes the favourable case where there is no accuracy problem (*i.e.* $\Delta_{\mathbb{C}_{\max}}(\sigma'_o.o) = \Delta_{Mod}(\sigma_o.o)$ = F-ambiguous).

Property 2. If $\Delta_{Mod}(\sigma_o.o)$ = F-sure and $\Delta_{\mathbb{C}_{\max}}(\sigma'_o.o)$ = F-ambiguous then there exists in $\mathbb{C}_{\max}$ at least two traces τ_F ($F \in \tau_F$) and $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) such that :

- $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'.o$,
- $\forall \tau \in \mathcal{T}(\mathbb{C}_{\max}) | \, \mathrm{P}_{\Sigma_o}(\tau) = \sigma'.o \wedge \mathrm{P}_{\Sigma_r^{ext}}(\tau) = \mathrm{P}_{\Sigma_r^{ext}}(\tau_F) \implies F \in \tau$.

Proof : The first condition stems from the fact that $\Delta_{\mathbb{C}_{\max}}(\sigma'_o.o)$ = F-ambiguous if and only if there exists at least two traces τ_F ($F \in \tau_F$) and $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) such that $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'_o.o$. The second condition directly stems from the property 1 by contraposition and that allows for the fact that $\Delta_{Mod}(\sigma_o.o)$ can be F-sure. □

Property 2 states that accuracy problems come from both the presence of local faulty and non-faulty traces that emit the same observable sequence but do not interact with the neighbourhood of $\mathbb{C}_{\max}$ in the same manner (second condition of Property 2). Hence the following result, if such a problem occurs a finite number of time, the local diagnoser of $\mathbb{C}_{\max}$ is accurate.

Property 3. For $\Delta_{\mathbb{C}_{\max}}$ to be accurate, it is sufficient that the set of couples $(\tau_F, \tau_{\neg F})$ defined by property 2 is finite.

Proof : Consider an observable sequence $\sigma_o.o$ with o emitted from $\mathbb{C}_{\max}$ such that $\Delta_{Mod}(\sigma_o.o)$ = F-sure and let us suppose that $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma.o))$ = F-ambiguous. If $\Delta_{\mathbb{C}_{\max}}$ is not accurate, then there exists at least one finite suite of observable continuations $\sigma_{o1}o_1$, $\sigma_{o1}o_1\sigma_{o2}o_2 \ldots$ with o_i emitted from $\mathbb{C}_{\max}$ such that $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma_o o \sigma_{o1}o_1))$ = F-ambiguous, $\Delta_{\mathbb{C}_{\max}}(\mathrm{P}_{\Sigma \to \Sigma_o}(\sigma_o o \sigma_{o1}o_1\sigma_{o2}o_2))$ = F-ambiguous ... hence the presence of an infinite set of couples $(\tau_F, \tau_{\neg F})$ according to property 2. □

Detection criterion of the accuracy of $\mathbb{C}$

The difference between any configuration $\mathbb{C}$ and the associated configuration $\mathbb{C}_{\max}$ is the relaxation of internal connections that leads the diagnoser $\Delta_{\mathbb{C}}$ to consider a set of traces that contains the set of traces of $\mathbb{C}_{\max}$. The consequence in terms of accuracy is the following. Given that $\Delta_{Mod}(\sigma_o.o)$ = F-sure, there necessarily exists a trace $\tau_F \in \mathcal{T}(\mathbb{C}_{\max}, \neg F)$ containing F that forms part of the explanation of $\sigma_o.o$ as described previously. Where $\mathbb{C}$ is concerned, the diagnoser $\Delta_{\mathbb{C}}$ answers not only in terms of the presence or absence of traces $\tau_{\neg F}$ of $\mathcal{T}(\mathbb{C}_{\max}, \neg F)$, producing the same observations as τ_F, but also in terms of the traces $\tau_{\neg F}$ of $\mathcal{T}(\mathbb{C}, \neg F) \setminus \mathcal{T}(\mathbb{C}_{\max}, \neg F)$ producing the same observations but coming from the relaxation of internal connections of $\mathbb{C}$. The consequence is an accuracy criterion for $\Delta_{\mathbb{C}}$ that is identical to that for a configuration $\mathbb{C}_{\max}$ (*i.e.* the property 3) but relies on the following property 4 that extends property 2. This leads to definition 46 for the accuracy criterion that applies to any configuration $\mathbb{C}$.

Property 4. If $\Delta_{Mod}(\sigma_o.o)$ = F-sure and $\Delta_{\mathbb{C}}(\sigma'_o.o)$ = F-ambiguous then there exists in $\mathbb{C}_{\max}$ at least one trace τ_F ($F \in \tau_F$) and in $\mathbb{C}$ one trace $\tau_{\neg F}$ ($F \notin \tau_{\neg F}$) such that :

- $\mathrm{P}_{\Sigma_o}(\tau_F) = \mathrm{P}_{\Sigma_o}(\tau_{\neg F}) = \sigma'_o.o$,

- $\forall \tau \in \mathcal{T}(\mathbb{C}_{\max}) | \, \mathrm{P}_{\Sigma_o}(\tau) = \sigma'_o.o \wedge \mathrm{P}_{\Sigma_r^{ext}}(\tau) = \mathrm{P}_{\Sigma_r^{ext}}(\tau_F) \implies F \in \tau.$

Definition 46 (Accuracy Criterion)**.** For the diagnoser $\Delta_{\mathbb{C}}$ of any configuration $\mathbb{C}$ to be accurate, it is sufficient that the set of couples $(\tau_F, \tau_{\neg F})$ defined by property 4 is finite.

5.4.5 Verification Algorithm

We are now ready to describe an algorithm that checks whether $\Delta_{\mathbb{C}}$ is accurate or not which relies on the properties described in section 5.4.4. The first remark is that the diagnoser $\Delta_{\mathbb{C}}$ is only accurate if the diagnoser $\Delta_{\mathbb{C}_{\max}}$ is itself accurate (this comes directly from the definition). The proposed algorithm is described in terms of languages and successively analyses the accuracy of $\Delta_{\mathbb{C}_{\max}}$, then of $\Delta_{\mathbb{C}}$. By consecutive operations of intersection and projection of languages, the algorithm eliminates the traces that do not lead to a problem of accuracy and retains at the end only traces that present problems. If this number of traces is finite, we conclude that $\Delta_{\mathbb{C}}$ is accurate. As stated by properties 2-4, only interactive events Σ_r^{ext} and observable events Σ_o come into consideration in the verification of accuracy. The other type of events are abstracted by projection of traces $\mathcal{T}(F)$ and $\mathcal{T}(\neg F)$ (lines 2–3). With lines 4–5, the objective is to calculate the sources of ambiguity that do not present a problem of accuracy (see property 1), by the intersection $\mathcal{T}(F) \cap \mathcal{T}(\neg F)$ and can thus be eliminated from $\mathcal{T}'(F)$. Then, the algorithm checks that there does not exist in the remaining traces of $\mathcal{T}'(F)$ an infinite set of traces (loop detection) whose observable projection is also the same as that of traces coming from $\mathcal{T}(\neg F)$ (line 6). To this end, we calculate the set of observable projections common to $\mathcal{T}(F)$ and $\mathcal{T}(\neg F)$ and by inverse projection find the traces of $\mathcal{T}'(F)$ to preserve. Finally, if $\mathcal{T}'(F)$ is finite (lines 7–11) then property 3 is verified and $\Delta_{\mathbb{C}_{\max}}$ is accurate. It is sufficient to iterate through the process (lines 12–20) to deal with the non-faulty traces of $\mathcal{L}_{\mathbb{C}}(\neg F) \setminus \mathcal{L}_{\mathbb{C}_{\max}}(\neg F)$ and compare them with the faulty traces of $\mathcal{L}_{\mathbb{C}_{\max}}(F)$ in order to establish if the extension of property 3 is also verified.

5.5 Illustrative Example

Going back to the example in Figure 5.1, if $\mathbb{C}$ only contains component A, then $\mathbb{C} = \mathbb{C}_{\max}$. In this case, $\Delta_{\mathbb{C}}$ still returns an ambiguous result. There exists an infinite number of traces for which the fault F has occurred and whose interactive behaviour is different from that of traces for which F has not occurred (these traces begin with $\{F\}.\{a3\}.\cdots$ or $\{F\}.\{a2\}.\cdots$), thus property 2 is verified an infinite number of times. Note also that the other traces of F beginning with $\{F\}.\{a1\}.\cdots$ have the same interactive and observable projection as the traces in which F has not occurred. These traces are intrinsically ambiguous (see line 4 of algorithm 3). We now consider the sub-configuration $\mathbb{C} = \{\{A, B, C, D\}, \{\langle a_2, b_2\rangle, \langle a_4, b_4\rangle, \langle a_5, b_5\rangle\}, \{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}\}$ (see Figure 4). $\mathbb{C}_{\max}$ is necessarily accurate here since $\mathbb{C}_{\max}$ is the complete system in this simple example, $\Sigma_r^{ext} = \emptyset$, and in this case $\mathcal{T}'(F)$ (line 7) that results from this algorithm is empty by construction. It remains to check if the relaxation of connections $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$ induces an accuracy problem.

Algorithm 3 Verification of the accuracy of $\Delta_{\mathbb{C}}$

1: **Input :** Sub-configuration $\mathbb{C}$, Fault F
2: $\mathcal{T}(F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}_{\max}}(F))$
3: $\mathcal{T}(\neg F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}_{\max}}(\neg F))$
4: $Ambiguous(F) \leftarrow \mathcal{T}(F) \cap \mathcal{T}(\neg F)$
5: $\mathcal{T}'(F) \leftarrow \mathcal{T}(F) \setminus (Ambiguous(F))$
6: $\mathcal{T}'(F) \leftarrow \mathcal{T}'(F) \cap$
$\mathrm{P}^{-1}_{\Sigma_o \cup \Sigma_r^{ext}}(\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(F)) \cap$
$\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(\neg F)))$
7: **if** $\mathcal{T}'(F)$ is finite **then**
8: {Property 2 does not occur indefinitely.}
9: **if** $\mathbb{C} = \mathbb{C}_{\max}$ **then**
10: The diagnoser $\Delta_{\mathbb{C}}$ is accurate
11: **else**
12: $\mathcal{T}'(\neg F) \leftarrow \mathrm{P}_{\Sigma \to \Sigma_o \cup \Sigma_r^{ext}}(\mathcal{L}_{\mathbb{C}}(\neg F) \setminus$
$\mathcal{L}_{\mathbb{C}_{\max}}(\neg F))$
13: $\mathcal{T}'(F) \leftarrow \mathcal{T}(F) \setminus (Ambiguous(F))$
14: $\mathcal{T}'(F) \leftarrow \mathcal{T}'(F) \cap$
$\mathrm{P}^{-1}_{\Sigma_o \cup \Sigma_r^{ext}}(\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}(F)) \cap$
$\mathrm{P}_{\Sigma_o \cup \Sigma_r^{ext} \to \Sigma_o}(\mathcal{T}'(\neg F)))$
15: **if** $\mathcal{T}'(F)$ is finite **then**
16: {Property 4 does not occur.}
17: The diagnoser $\Delta_{\mathbb{C}}$ is accurate
18: **else**
19: Problem of inaccuracy of $\Delta_{\mathbb{C}}$ to occur due to relaxed internal connections
20: **else**
21: The accuracy of $\Delta_{\mathbb{C}}$ cannot be demonstrated at this stage

It then becomes sufficient to note that the relaxation does not cause an increase in the number of observable traces of $\neg F$ and therefore that the remaining set $\mathcal{T}'(F)$ is also empty (line 15). Relaxing connections $\{\langle a_1, b_1\rangle, \langle a_3, b_3\rangle\}$ is thus useful as $\Delta_{\mathbb{C}}$ is accurate.

5.6 Choosing a Sub-configuration

We discuss now how to choose a sub-configuration minimizing the cost of diagnosis (defined by the tree width) while ensuring an accurate diagnosis. A sub-topology $\mathbb{T}$ is *better* than another topology $\mathbb{T}'$ if the accuracy associated with $\mathbb{T}$ is stronger than the accuracy associated with $\mathbb{T}'$, or both accuracies are identical but the tree width of $\mathbb{T}$ is smaller than in $\mathbb{T}'$. To find an optimal sub-topology, we have to explore the set of sub-topologies $\Upsilon = 2^{\mathcal{K}}$ defined as the power set of the connections $\mathcal{K}$ in the system. The partially-ordered set $\langle \Upsilon, \subseteq, \supseteq \rangle$ forms a lattice. The cost and the accuracy have monotonicity properties in this lattice. Indeed, if $\mathbb{T} \subseteq \mathbb{T}'$, then

- since $\mathbb{T}'$ is a sub-topology of $\mathbb{T}$, the diagnosis with $\mathbb{T}$ is equal to or more accurate than the one with $\mathbb{T}'$, and

- any junction tree of $\mathbb{T}$ is also a junction tree of $\mathbb{T}'$, and the tree width of $\mathbb{T}'$ is equal to or smaller than that of $\mathbb{T}$.

These properties allow for an efficient search in Υ. A possible approach is to start from the physical topology and to remove connections as long as the accuracy is not affected. We work in a context with a huge number of components – possibly thousands – and expect to build sub-topologies with very small tree width – several units at most. Therefore, we recommend to instead start from the empty topology $\mathbb{T}_\perp = \emptyset$ and incrementally add connections; when an accurate sub-topology was determined, it is possible to refine it by removing connections that were added unnecessarily. This is illustrated Algorithm 4.

Algorithm 4 Exploration of Υ

1: **Input:** Γ, F
2: $\mathbb{T} := \emptyset$
3: **while** $\mathbb{T}$ is not accurate, **do**
4: Add a connection to $\mathbb{T}$.
5: **while** $\exists c \in \mathbb{T}$ s.t. $\mathbb{T} \setminus \{c\}$ is accurate, **do**
6: Remove c in $\mathbb{T}$.
7: **Return** $\mathbb{T}$

It is possible to improve the exploration of Υ as follows:

- The accuracy testing may generate an explanation for non accuracy, and indicate which connections of $\mathcal{K} \setminus \mathbb{T}$ are responsible for non accuracy. The connection added in line 4 may be chosen in this set of connections.

- When the junction tree of $\mathbb{T}$ is also a junction tree for $\mathbb{T}' \supseteq \mathbb{T}$, it is possible to test the accuracy immediately on $\mathbb{T}'$ since the cost associated with $\mathbb{T}$ and $\mathbb{T}'$ are identical.

The algorithm proposed here leads to a local optimum if the accuracy testing (line 3) is correct; if the condition for accuracy is sufficient but not necessary, the result may be not locally optimal.

5.7 Conclusion

This chapter proposes an original approach to reduce the complexity of the diagnosis of large discrete event systems by ignoring some connections in the system. Our work can be seen as a particular case of abstraction, similar to what is presented in [89]. It complements the work presented in chapter 4.

We note that to choose a sub-configuration minimising the cost of diagnosis, we do not put a bound on the delay required for the sub-configuration to become accurate. The fairness assumption allows us to predict that in most cases it will be a reasonable delay, *i.e.* a finite and limited delay relative to the number of components considered and physical assumptions imposed by the nature of the system. A way around the problem is to introduce a bounded delay in the definition of

accuracy. This would imply that the sub-configuration to choose might be bigger, thus needing a trade-off between sub-configuration size and delay. This is part of future extensions.

Other future work includes refining the cost function with additional factors such as the total number of clusters in the junction tree, the tree shape, the proportion of observable events in nodes, the longest line in the tree, etc. The accuracy criterion could also be improved, with the boolean result replaced by a real value that could allow for a trade-off between accuracy and cost. Another interesting improvement is to consider several faults. Each fault can be diagnosed by a junction tree, but those trees may include identical nodes. The question is then how to combine these trees.

Part II

Hybrid System Diagnosis

This part of the thesis, consisting of three chapters, presents an approach to the integration of DX and FDI methodologies. Instead of considering only the discrete evolution of systems, we also take into account and retain some of the continuous aspects of system evolution in the system model. This makes it possible to use methodologies for both the discrete and continuous operation of the system for diagnosis. We thus have a *hybrid model* of systems and can apply *hybrid methodologies* to them. Chapter 7 presents a framework for modelling hybrid systems and networks. Chapter 8 then explores diagnosability on hybrid systems.

Work presented in this part of the thesis was done in collaboration with Lachlan Blackhall, a fellow PhD student at the time. Lachlan was pursuing his PhD in the area of Control Theory. Lachlan contributed the Control Theory perspective to our joint work while I brought in the Diagnosis (in the DX sense) perspective. The algorithms for the estimation of structural and parametric changes have been developed by Lachlan as part of his thesis and are not covered in detail here although references are provided to his publications. These algorithms were extended and integrated within a hybrid framework to produce our joint work.

Chapter 6

Part II: Background

6.1 Motivation

We consider *hybrid dynamical systems* (for short we will use the term *'hybrid systems'* as well) in this section as dynamical systems that have both continuous and discrete dynamics. There has been emergent interest in such systems because modeling in the hybrid system formalism has the advantage of keeping information contained in the continuous dynamics of the system, while handling the discrete aspects at the same time. This will be further explained in the ensuing chapters.

The hybrid system framework unifies two distinct communities that have both been working on model-based diagnosis; namely the Fault Detection and Isolation (FDI) community and the Diagnosis (DX) community. The FDI community has evolved in the field of automatic control and uses techniques from systems and control theory. In comparison, the DX community has a more recent emergence with foundations in the field of computer science. In both approaches, system models and observations are used to determine the behaviour of systems. However, the nature of the models and the tools and techniques to handle them vary greatly. Generally, the FDI approach makes use of continuous analytical models and tools from linear algebra whereas the DX approach employs symbolic/qualitative models and tools from logic. The two approaches can be linked as shown in [28] and when integrated can offer more powerful capabilities for fault detection.

By defining a hybrid system framework, we want to bring together methodologies from both fields for handling the diagnosis of hybrid dynamical systems. Systems with complex dynamics are hard to track using FDI techniques alone. Hence abstracting a discrete layer which models the system on a qualitative level allows us to use reasoning techniques, layered on top of the continuous dynamics, that are able to handle the complexity. The consistency-based tools from DX work well for systems where fault symptoms are clearly distinguishable from nominal behaviour. Signal noise is rarely a consideration in the latter approach which deals with qualitative models and hence it is often assumed that fault symptoms can be identified without ambiguity. However, often,

fault symptoms get masked with the presence of signal noise until the effects become catastrophic. For example, while the the fault causing the loss of Mars Polar Lander was a software bug, the fault symptoms were not detected early enough as they were masked by noise until the situation was critical. To tackle this problem, as the authors in [99] mentioned, there are two main issues to address. The first one is that faults manifest through a combination of continuous dynamics and discrete mode changes. Hence hybrid monitoring and diagnosis capabilities are required to be able to track a system's evolution on those two levels. The second issue is that faults may generate symptoms of the same order of magnitude as sensor and actuator noise. To be able to detect these symptoms, statistical methods need to be applied to separate the noise from the true dynamics.

In this part of the thesis, a hybrid system formalism and associated techniques are presented. We introduce the notion of a *hybrid dynamical network* which consists of many components (each of which is a hybrid system in itself) that are physically linked together. The network displays an overall complex behaviour, despite the fact that the behaviour of its constituent components could be simple, because the components affect one another. In chapter 7, we present a framework for modelling hybrid dynamical systems and networks. We then present diagnosability results on networks of interconnected system using *indicator functions* in chapter 8. The methodology presented focusses on detecting operational mode changes that can be determined by changes in the fundamental governing dynamics of the system.

6.2 Hybrid System Framework

A hybrid system model encompasses both the continuous dynamical behaviour of the system and the evolution of the system through different operational modes. We present in Table 6.1 ambiguous terms that are used in both fields in different ways and clarify how we are going to use them in our context.

Controls (FDI)	**Diagnosis (DX)**	**Hybrid (our framework)**
state	-	state
mode	state	mode
-	trajectory	mode trajectory
trajectory	-	state trajectory
-	event	event
changepoint	transition	transition
system	component	system
network	system	network

Table 6.1: FDI and DX cross-field translation.

A mode in this context is what is usually considered to be a state (*e.g.* on an automaton) in the discrete event system sense. A trajectory in the FDI literature refers to a *state trajectory*

(as given in definition 7, chapter 2) which is the set of all possible values that the continuous state vector $\mathbf{x}$ can take from given initial conditions in the solution space of a particular set of dynamical equations. However in the DX literature, a trajectory refers to a path on an automaton (as given in definition 16, chapter 2) that links an initial state (in the DX sense) to a final state. Given that we are calling a state of an automaton a *mode* in this part of the thesis, we can refer to the DX trajectory as a *mode trajectory* which is the set of all possible discrete modes that the system can go through from a starting initial mode. An *event* refers to a discrete event occurring in the system that causes a change from one discrete mode to another, *e.g.* in an electricity network, it could be a fault on a power line. We call the change from one discrete mode to another a 'transition'. A transition is usually triggered by either a control input or changes happening in the system. In FDI jargon, a transition corresponds to a *changepoint*, which describes an abrupt variability in data caused by a change in the operating dynamics of the system.

6.3 Hybrid Systems

An introduction to hybrid systems was given in chapter 2 in relation to previous work done in the field. In this section, we define a hybrid dynamical system formally as relevant to the work presented to this part of the thesis. A system is called *dynamical* when continuous dynamics are used to represent its evolution.

Definition 47 (Hybrid Dynamical System)**.** A hybrid dynamical system is a tuple $\langle \mathcal{M}, \Omega, \mathcal{T}, \mathbf{x}, \mathbf{y}, \mathbf{u}, \Theta, \mathcal{F}, \mathcal{H} \rangle$

- The finite set $\mathcal{M}$ denotes the modes $m_\tau \in \mathcal{M}$ of the system
- Ω denotes the set of possible events in the system that could cause a change in mode through a transition in $\mathcal{T}$ where $\mathcal{T}$ is a function that maps an event and mode into a new mode ($\mathcal{T} : \mathcal{M} \times \Omega \rightarrow \mathcal{M}$).
- $\mathbf{x}$ and $\mathbf{y}$ are respectively the continuous state variables vector and the output variables vector. Elements of both vectors range over $\mathbb{R}$.
- $\mathbf{u}$ is the vector of continuous control input variables, which also range over $\mathbb{R}$.
- The set Θ associates with each mode $m_\tau \in \mathcal{M}$ a set $\boldsymbol{\theta}_\tau$ of system parameters. The cardinality of this set and values of the elements depends on the specific system being looked at.
- The sets $\mathcal{F}$ and $\mathcal{H}$ associate with each mode $m_\tau \in \mathcal{M}$, continuous functions f_τ and h_τ respectively, which together with $\boldsymbol{\theta}_\tau$ completely describes the continuous dynamics exhibited at m_τ by equation 6.1.

$$\begin{aligned} \dot{\mathbf{x}} &= f_\tau(\mathbf{x}, \mathbf{u}_\tau, \boldsymbol{\theta}_\tau) \\ \mathbf{y} &= h_\tau(\mathbf{x}) \end{aligned} \tag{6.1}$$

The derivative of $\mathbf{x}$ with respect to time is denoted $\dot{\mathbf{x}}$. We write τ as index of the operation mode for simplicity but there is an implicit time dependence, *i.e.* $\tau(t)$. Hence mode changes could be used to denote 'logical time' which is the abstracted time scale used for reasoning at the logical level.

The system starts in an initial operational mode which can change over time. Each mode change has an associated time of occurrence which is not necessarily known nor observed directly. Hence we need some form of diagnosis method to determine if a change has happened.

A *hybrid dynamical network* consists of interconnected hybrid dynamical systems. Diagnosis on such a network is explored in details in chapter 7.

Our hybrid system approach differs from those in [12, 73, 25] in that the occurrence of events, or equivalently of mode changes, is measured not through residual estimation but through changes in the governing dynamics. Rather than defining guard functions for the transitions as in [3, 64, 47], changepoint detection (a technique used on the continuous dynamics) is used to determine a change from one mode to another. This is explained further in section 7.3. There are also no specific reset functions defined on the system modes as there might be modes from which it is not possible to recover (as in the example transformer system presented in section 6.4). The hybrid systems under consideration can thus be viewed as *exogenous switching systems*.

6.4 Illustration of Hybrid Systems

We illustrate what we mean by hybrid systems by walking through a simplified version of the operation of a component of an electricity network, namely a substation. The relationship between continuous dynamics and discrete mode, as considered in the framework used in this work, will be made clear. We consider a substation operating as part of an electricity network. For example, the substation highlighted and circled in figure 6.1. The main function of a substation is to convert voltage from one value to another, which is done through a *transformer*. Voltage either needs to be stepped up (from a lower to a higher value) or down (from a higher to a lower value). A *transformer* is a device that converts electrical energy from one circuit to another without direct electrical connection [37]. It normally consists of a magnetisable metal core around which two sets of coils are wound, as shown in figure 6.2. The transformer works on two basic principles. The first one is that an electric current can produce a magnetic field. The second is that a changing magnetic field within a coil of wire induces an electromotive force (emf) across the ends of the coil. Emf is the external work expended per unit of charge to produce an electric potential difference (or voltage) across two open-circuited terminals. The created electrical potential difference drives current flow if a circuit is attached to the source of emf. Emf and potential difference are covered in any basic physics textbook or course notes on electricity and the reader is referred to resources such as [80] or [31].

The continuous dynamics of a transformer is thus governed by Faraday's Law of electromagnetic induction. It states that the rate of change of magnetic flux (with respect to time) is

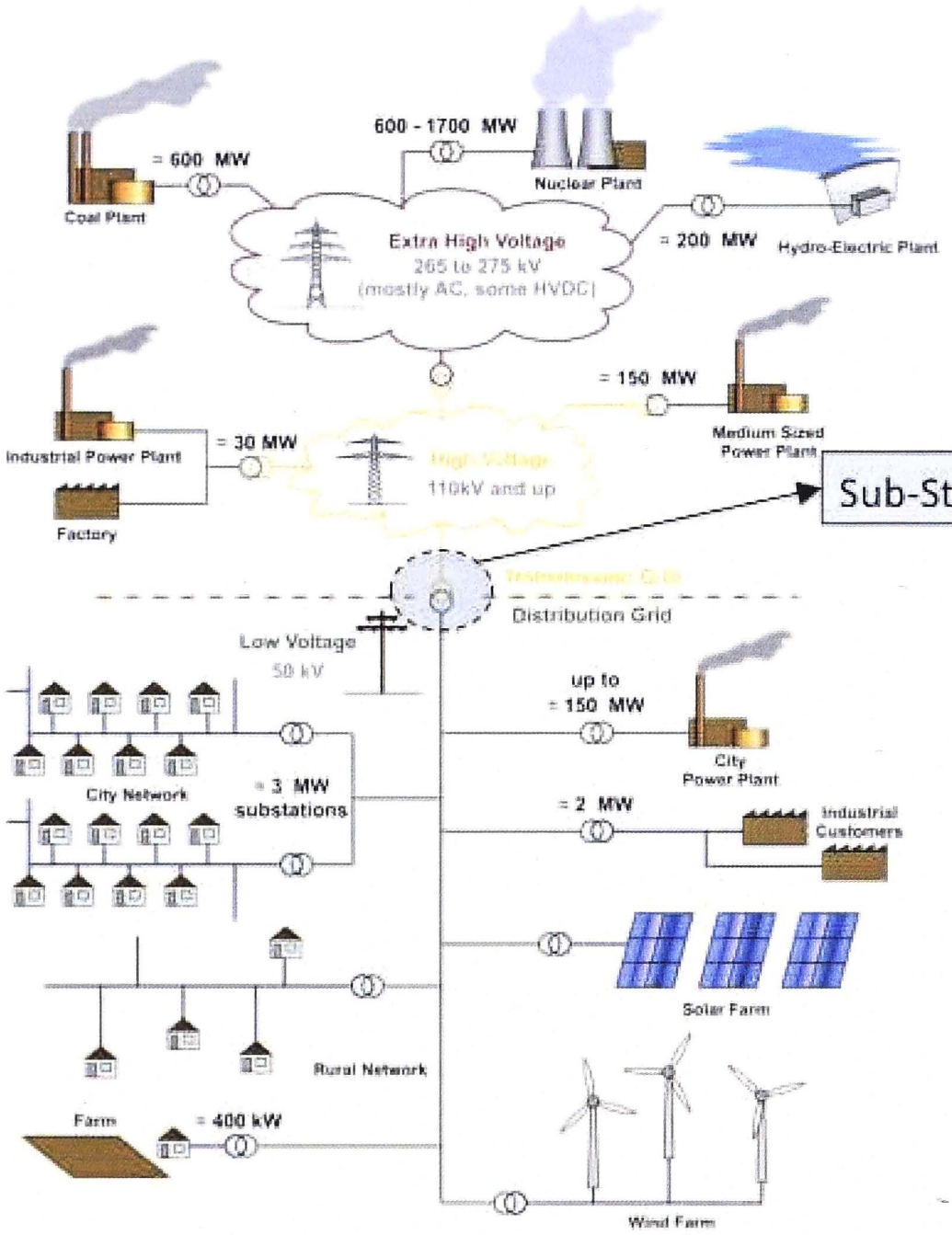


Figure 6.1: Electricity Network (image licensed under the Creative Commons Attribution 3.0 Unported license, obtained from [8])

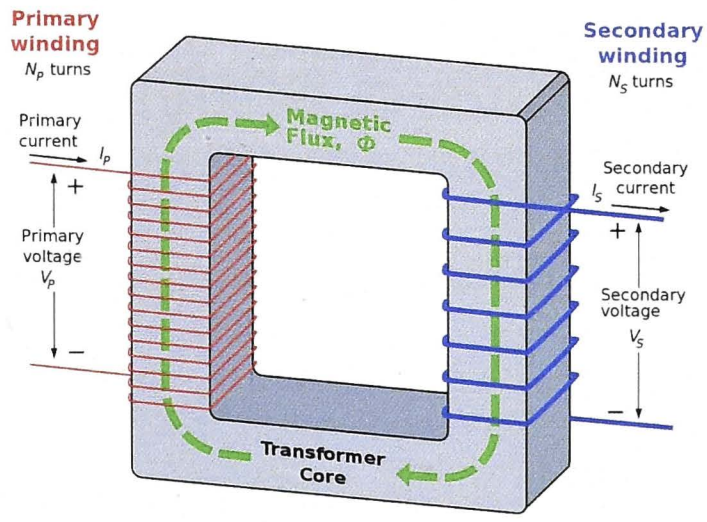


Figure 6.2: Transformer: Main electrical device used at a substation(image under GNU Free Documentation License, obtained from [7])

proportional to the emf induced in a conductor or coil. In fact, the voltage V_s induced across the secondary coil is given by:

$$V_s = emf_s = N_s \frac{d\Phi}{dt} \tag{6.2}$$

where Φ is the magnetic flux induced and N_s is the number of turns in the secondary coil.

During a significant power failure in a power network, two unusual operating conditions can most significantly affect the operating mode of a transformer [74]:

1. Underfrequency operation

2. Overvoltage operation

In the underfrequency mode, the load of the network or subnetwork exceeds generation resulting in reduced frequency of the voltage and current.

In the overvoltage mode, the transformer is subjected to an abnormally high voltage which can be due to load rejection or lightning surge, or the system being pieced back together after being split apart because of a disturbance.

Both conditions can lead to overheating of the transformer and might cause fires and explosions in the worst case. But even in a less drastic scenario, there can be a huge cost associated with transformer failure due to loss of generation revenue and cost of damage repairs.

The timeframe before failure in both conditions is relatively short (in the order of minutes) [2]. Hence it is important to have an automatic detection and protection mechanism to quickly recognise abnormal conditions and take actions to prevent damage to the transformer.

In a hybrid system context, as illustrated in figure 6.3, the continuous operation of the transformer can be abstracted into four discrete modes (using the particular conditions described in this section): Normal Mode, Underfrequency Mode, Overvoltage Mode and Failure Mode. A Hybrid Automaton showing these four discrete modes is shown in figure 6.4.

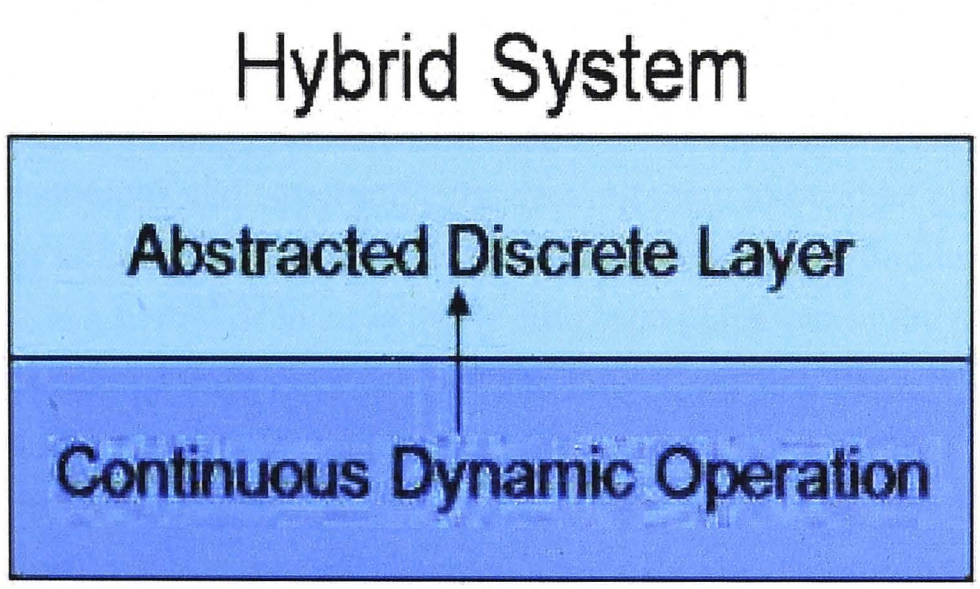


Figure 6.3: Hybrid System Abstraction

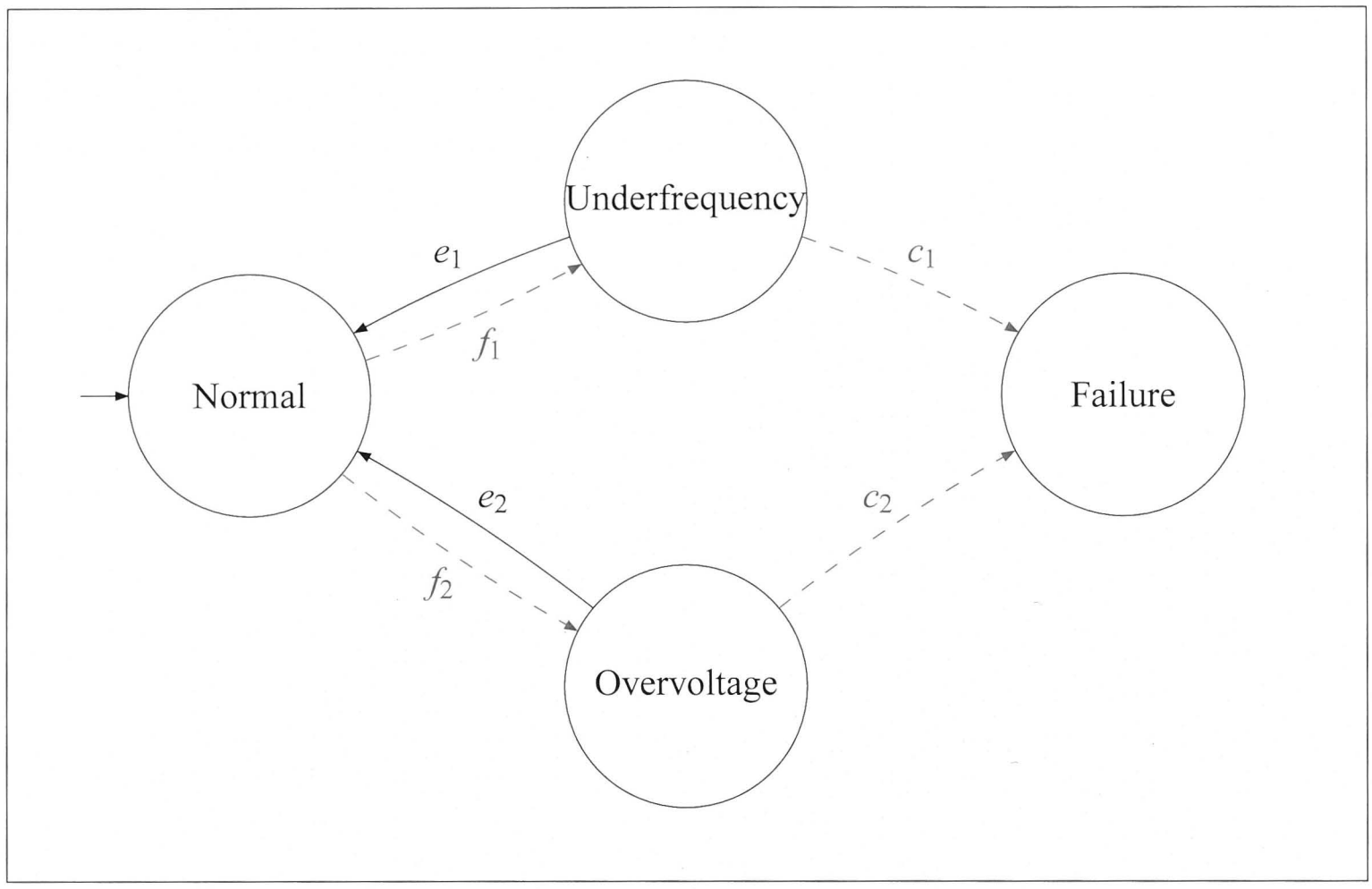


Figure 6.4: Automaton representing a transformer as a hybrid system with different discrete modes of operation

In normal mode, the alternating current current and voltage applied to the primary coil are within safe range. Magnetic flux is induced in the metal core which in turn induces a voltage across the secondary coil. The voltage induced in the secondary coil is also within safe range. The transformer operates as expected.

According to Faraday's law of electromagnetic induction, flux in the transformer core is directly proportional to voltage and inversely proportional to frequency. This means that both underfrequency fault f_1 and overvoltage fault f_2 have a similar effect on a transformer, known as *overexcitation*. Under these conditions, more flux that the core steel can handle is generated (*i.e.*core saturation is reached). Overflow flux strays outside of the core steel and can link up conducting loops in windings and structural parts causing them to conduct electricity and to overheat. If the situation is not rectified within a reasonable time limit, this leads to failure of the transformer resulting in a partial or in an extreme case total black-out in the power network.

When an underfrequency fault f_1 or overvoltage fault f_2 occurs, there is a short timeframe in which repair actions e_1 or e_2 can be undertaken. Beyond this critical timeframe for action, the transformer enters failure mode where it will need to be shut down and taken off the network. The events c_1 (Underfrequency mode was not detected and/or Repair was not initiated on time) and c_2 (Overvoltage mode was not detected and/or Repair was not initiated on time) represent this shift of operation to failure mode.

It is to be noted that the discrete modes of Underfrequency and Overvoltage can be further ab-

stracted into one encompassing Overexcitation mode. However, information, which coud potentially help in more efficient detection and repair, would be lost. Underfrequency and Overvoltage modes could be detected using relay circuits like in [2] or vibroacoustic signals like in [11].

Chapter 7

Diagnosis of Hybrid Dynamical Systems for Fault Tolerant Control

7.1 Introduction

Diagnosing and controlling hybrid dynamical systems pose many challenges. These challenges are somewhat more significant in the presence of faults. Faults drastically alter the fundamental governing dynamics of the system rendering the original, fault-free control methodology severely degraded or useless.

We thus require that an effective controller be able to detect that a fault has occurred and change control strategy accordingly. Choosing an appropriate control strategy is achievable and has been the focus of extensive work in the field of control theory [61, 1]. Additionally these control strategies can usually be computed off-line. What is lacking is a systematic on-line methodology to automate the process of detecting the occurrence of a fault(s) and choosing the most appropriate control strategy. Additionally we wish to minimise the fault monitoring required to still achieve the global performance objectives. Allowing the fault detection and recovery strategy to run in a distributed manner would be an additional benefit.

As mentioned in chapter 2, much of previous work focuses on detecting faults by measuring deviations of a hybrid dynamical system from the ideal or nominal trajectory, *i.e.* using residuals. Residual generation for fault detection requires advance knowledge of the system operation and parameter values. This approach necessitates that the structure and parameters of the underlying dynamics are known apriori. *i.e.* the system must be very well characterised. Additionally any explicit parameters in the system dynamics definition must not drift (undergo deviations from their original value) during the operation of the hybrid system.

As hybrid systems become increasingly more complex, using the residual generation approach for fault detection becomes more challenging as the notion of a nominal trajectory is often hard to define in this context. To overcome these challenges, the diagnosis methodology we propose in this chapter focusses on detecting operational mode changes that can be determined by changes in the

fundamental governing dynamics of the network, thus not requiring the computation of residuals. We use methods from estimation and regression to determine the fundamental dynamics before using changepoint detection to detect the occurrence of arbitrary events in the hybrid system. This represents a diagnosis on the continuous dynamics but by then applying existing discrete diagnosis methods, this diagnosis can be refined to produce a hybrid systems diagnosis that unifies methods from both the FDI and DX communities. We present results on a hybrid system with chaotic oscillator modes and on a four-node hybrid dynamical network, both exhibiting complex behaviour.

7.2 Preliminaries

We consider *hybrid systems* as introduced in Chapter 6. Hybrid systems are characterised by having a finite number of discrete operating modes with different dynamic operations within each mode. We use the definition provided in section 6.3. The governing dynamics of the system is given by equation 6.1, reproduced here:

$$\dot{\mathbf{x}} = f_\tau(\mathbf{x}, \mathbf{u}_\tau, \theta_\tau)$$
$$\mathbf{y} = h_\tau(\mathbf{x})$$

where τ defines the operating mode and takes values in $M = \{1, 2, \cdots, m\}$. It is assumed that the operating mode is known at time t_0.

7.2.1 Switching sequence and switching signal

We assume that the state of a hybrid system is continuous and thus does not exhibit abrupt changes at the instant of changing operating mode.

Definition 48 (Switching sequence)**.** A switching sequence Ψ is a sequence that describes the sequence of mode changes with the evolution of time for a given hybrid system.

$$\Psi = \{(i_0, t_0), (i_1, t_1), \cdots, (i_z, t_z) | i_z \in M, z \in \mathbb{Z}\}$$

where i_0, t_0 are the initial system modes and time respectively and $\mathbb{Z}$ is the set of nonnegative integers. When $t \in [t_z, t_{z+1}), \tau = i_z$, we say the i_z-th operational mode is active.

Definition 49 (Switching signal)**.** Given a switching sequence Ψ, as defined in definition 48. When the i_z-th operational mode is active, we say the trajectory of the hybrid system ($\mathbf{x}(t)$) is defined as the trajectory of the system ($\mathbf{x}_i(t)$) with switching signal $i \in M$. The switching signal denotes a change in mode of the system.

We can think of the hybrid system being defined by its state, output and control vectors, and parameter vector $S = \langle \mathbf{x}, \mathbf{y}, \mathbf{u}_\tau, \theta_\tau \rangle$.

The continuous behaviour of the hybrid system can be described by the evolution of its output, its state and the influence of its control inputs which are governed respectively by the tuple $S_{cont} = \langle f_\tau, h_\tau \rangle$ where the elements of the tuple have been defined previously.

The discrete behaviour of the hybrid system can be represented using a deterministic finite automaton. Thus the automaton representing the discrete behaviour of a hybrid system can be given by the tuple $S_{disc} = \langle M, \Omega, \mathcal{T} \rangle$.

- M is the finite set of system modes ($\{1, 2, \cdots, m\}$).
- Ω is the set of events ($\omega_1, \omega_2, \cdots, \omega_l$) which determine switching between modes.
- $\mathcal{T}$ is the transition function that maps an event and mode into a new mode, $\mathcal{T} : \Omega \times M \rightarrow M$.

The operating modes in a hybrid system completely describe the operation of the system. These modes can correspond to arbitrary operating conditions, some nominal and some induced through the occurrence of a fault or other event. We use the term events to be very general and include faults and other events of interest that can be characterized by a change in operating mode of the system.

7.2.2 Structural and Parametric Variations in Hybrid Systems

In Eq. 6.1 we have both f_τ and θ_τ characterizing the dynamics of each operational mode determined by τ. It is well known that a dynamic system can undergo bifurcations. These bifurcations are caused by a change in the system parameters θ_τ not by any change in the form of the dynamic system itself as would be implied by a change in f_τ. In separating the parametric variations of the operating mode from the structural changes (those changes resulting in a fundamental change in the form of f_τ) it is possible to deal compactly with changes in the operational mode of the hybrid system.

By explicitly allowing parameter variations in a given operation mode we are more able to deal with the subtleties of fault detection as it is now possible to encompass both structural and parametric variations explicitly. It should be noted that current approaches to hybrid systems diagnosis require exact knowledge of parameters ahead of time, thus if a given fault were to induce a bifurcation both the nominal and bifurcation operating modes would need to be simulated, something not required in the approach that we will detail.

7.3 Methodology for Hybrid System Diagnosis

The goal of diagnosis becomes determining $\overline{\Psi}$ such that $\overline{\Psi} = \Psi$. That is we wish to determine the sequence of operating modes and the events that caused these operational mode changes for all operating time of the system.

In order to diagnose hybrid dynamical systems we will use a two stage approach that combines both FDI and DX techniques. As our observations come from the continuous domain it is necessary to abstract the continuous operation of the hybrid system into observations in the discrete domain. Our approach is to use estimation techniques to identify the underlying dynamics of the hybrid system, thus identifying the current operational mode. In this way we have abstracted the continuous trajectory measurements into discrete observations of the hybrid system mode.

It is then possible to use changepoint detection techniques to determine possible operational mode transitions that may have occurred in the hybrid system. These changepoints are indicative of events occurring. At this point we have a diagnosis of the hybrid system using only the continuous dynamics. Using the deterministic finite automaton finally allows us to determine if the transitions that emerged from the changepoint detection techniques are valid and if they are what fault has occurred. We now have a diagnosis that incorporates knowledge from both the continuous and discrete dynamics of the hybrid system.

For the analysis that follows we assume that between two observations only one event can occur. This further implies that for any finite $T > t_0$, there exists a positive integer K_T, which may depend on T, such that during the time interval $[t_0, T]$ the operational mode τ changes no more than K_T times. This ensures that it is practically possible to diagnose a system as the sequence of observations will have information about all the faults that occur in the system.

7.3.1 Estimation

It was mentioned in Section 7.2 that hybrid dynamical system have a number m of distinct operational modes. These operational modes correspond to different dynamical models, whereby each model has different governing dynamics. Each dynamical model can correspond to a nominal or faulty operating mode and no distinction is made about whether the transition between modes is intentional or due to a fault event occurring.

The estimation approach uses estimation and regression techniques to try and estimate directly the governing dynamics, and hence the current operational mode. This differs markedly from the residual generation techniques in works such as [12, 73, 25]. The fields of estimation and regression have considerable literature. Background material for both fields can be found in [69] and [5]. In our approach to estimation we will assume that the form of the dynamics, that is the function f_τ, is known and that we use estimation and regression techniques from [15, 16] to determine the parameter vector θ_τ which when coupled with the form of the dynamics in f_τ will provide complete knowledge of the underlying dynamics. The estimation of continuous dynamics is based on the use of a Gaussian Sum filter which can be considered as a weighted bank of Kalman filters operating in parallel, where the weight of each filter changes after each measurement is processed. More details are given in [15, 16].

This approach is more general than that of residual generation as the form of the dynamics can generally be determined, leaving the often unknown and variable parameters (θ_τ) to be estimated as the hybrid system is observed. That is $\forall i \in M$ we are assuming that f_τ is known and θ_τ is

unknown and must be estimated.

The goal of estimation thus becomes the problem of estimating the parameters $\boldsymbol{\theta}_\tau$ of each model and to then determine the likelihood that a given operating mode f_τ with parameters $\boldsymbol{\theta}_\tau$ is the best representative of the current operating mode. This is discussed in greater detail below.

Given m possible operating modes or dynamical models we wish to use discrete observations $Y_t = \{y_0, y_1, \cdots, y_t\}$ to determine:

$$P(i|Y_t), \quad \forall i \in M \tag{7.1}$$

for any t where we naturally require:

$$\sum_{i=1}^{m} P(i|Y_t) = 1 \tag{7.2}$$

This corresponds to determining the relative probabilities of each of the models, with their current parameter estimate, thereby giving an indication of the model most likely to be responsible for generating the current trajectory. This is related to current work in the field of sparse estimation ([16]) where the relative probability for competing models is an output of the estimator.

This approach may appear similar to the process of residual estimation but there is a subtle and important difference here. In this approach only the structure of a dynamical model is presumed. By using the estimation approach we are not requiring parameters to have a fixed or constant value and even parameter drift can be dealt with in a manner that will not lead to spurious fault generation. Having a record of these probabilities we will now discuss the process by which we can detect possible changepoints in the operating mode of the hybrid system, generating fault observations that will then be passed to the discrete diagnosis algorithms for a final diagnosis.

7.3.2 Changepoint Detection

Changepoint detection is the process whereby changes in the trajectory of a series of measurements can be related to a change in the generation of the trajectory. In our framework, changepoint detection corresponds to determining the time at which the operational mode generating the state trajectory changes. From this change it is then possible to determine the probable events that caused this operational mode change through knowledge of the automaton governing the operation of the discrete component of the hybrid system.

Changepoint detection can also be regarded as the model selection problem [41] and in this light, couples well with the estimation approach detailed in the previous section. Both [24] and [41] provide some background to changepoint detection using estimation approaches. Our approach differs slightly from the methods cited in that we are assuming that it is possible to measure the probability of a given model using observations from the continuous trajectory of the system. That is we assume that our estimation techniques (discussed in the previous section) will give

$$P(i|Y_t), \quad \forall i \in M \tag{7.3}$$

Given these relative probabilities it is possible to compute the transition probabilities between operating modes as

$$P(i \rightarrow j|t) = P(i|Y_{t-1})P(j|Y_t) \tag{7.4}$$

$\forall i, j \in M$. If $i = j$ we are computing the probability that the current operational mode remains the same. Using this approach we obtain m^2 transition probabilities. Selecting l transitions where $l \leq m^2$ then represents choosing the l most likely transitions (events) to have occurred. The chosen transitions represent the diagnosis of the hybrid system from observation of the continuous state trajectory of the hybrid system. These transitions are the candidate events that may have occurred and must be refined using a discrete diagnosis approach to achieve a full hybrid system diagnosis.

7.3.3 Discrete Diagnosis

So far, we have described how candidate events are determined from observations and analysis on the continuous operation of the hybrid system. As the next step, we wish to refine those estimates using discrete diagnosis methods. We are using model-based discrete diagnosis, where we are assuming a model of the discrete operation of the hybrid system (S_{disc}, introduced in Section 7.2) is available (*i.e.* given an operational mode, the model describes the possible transitions that will result in a new operational mode). Model-based diagnosis has been covered extensively in part I of this thesis. Essentially, considering the discrete evolution of the system, the set of all possible behaviours of the system is a language denoted *Mod* over the set of events Ω that could possibly occur on the system.

The mode estimates obtained from analysis on the continuous part of the model can be viewed as the *discrete observations* Obs that are used with the discrete model S_{disc} in order to obtain a discrete diagnosis. The diagnosis of the system can thus be computed as

$$\bar{\Sigma} = \text{Mod} \otimes \text{Obs}. \tag{7.5}$$

where $\otimes$ is the synchronisation of the model on the discrete observations (as covered in part I of this thesis) . The diagnosis $\bar{\Sigma}$ essentially determines the likely transitions consistent with the observations. In this way, we are able to refine the original mode estimates in Obs.

7.4 Example

We will use the proposed hybrid diagnosis methodology to diagnose the operation of a four mode hybrid system. We use chaotic oscillators with varying parameters to represent both bifurcations and structural changes in the hybrid system.

We use two types of chaotic oscillator, the Rossler and Lorenz attractors. The Rossler attractor

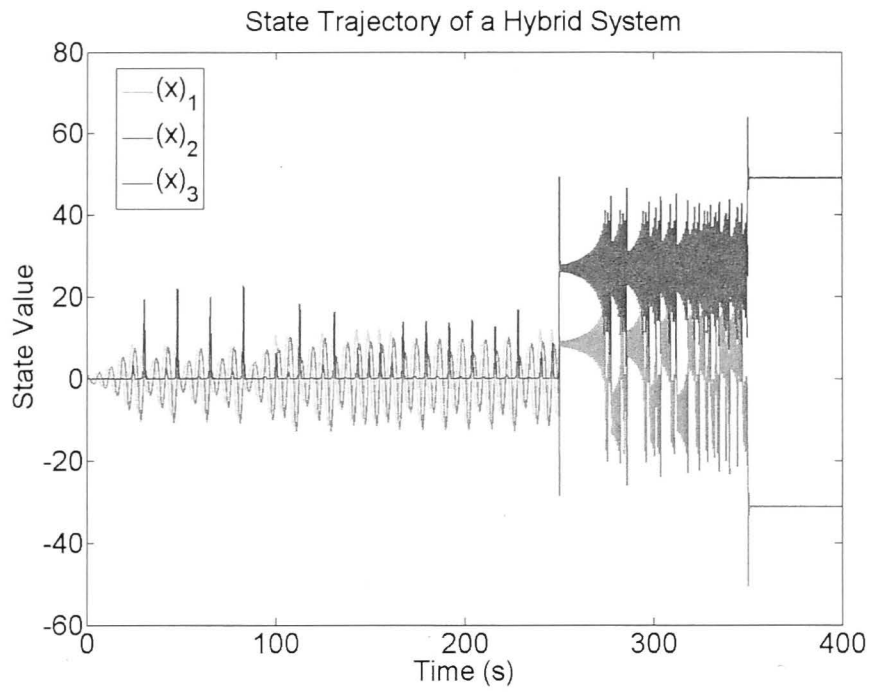


Figure 7.1: Rossler attractor parameter estimates.

(with three internal states) is parametrised by three parameters $\{p_1, p_2, p_3\}$ and is given by:

$$(\dot{x})_1 = -(x)_2 - (x)_3 \tag{7.6}$$

$$(\dot{x})_2 = (x)_1 + p_1(x)_2 \tag{7.7}$$

$$(\dot{x})_3 = p_2 + (x)_3((x)_1 - p_3) \tag{7.8}$$

The Lorenz attractor (with three internal states) is parametrised by five parameters $\{q_1, \cdots, q_5\}$ and is given by:

$$(\dot{x})_1 = q_1((x)_2 - (x)_1) \tag{7.9}$$

$$(\dot{x})_2 = q_2(x)_1 - q_3(x)_2 - (x)_1(x)_3 + q_4 \tag{7.10}$$

$$(\dot{x})_3 = (x)_1(x)_2 - q_5(x)_3 \tag{7.11}$$

Both of these attractors meets the definition of Eq. (6.1) as they have well defined dynamical structure and parameters that define the operation of the hybrid system. We assume that $y(t) = [(x)_1(t) \;\; (x)_2(t) \;\; (x)_3(t)]^T$ is the observable trajectory of the system. The hybrid system has the automaton shown in Fig. 7.2 and the true dynamics and parameters for each mode are shown in Table 7.1. We simulate the dynamics of the hybrid system in MATLAB for 400s using the ODE45 integrator. Mode changes are induced into the program with the times and corresponding events given in Table 7.2. Using this approach the trajectory of the hybrid system can be seen in Figure 7.1.

To estimate the parameters in both the Rossler and Lorenz attractors of Eq. (7.8) and (7.11) we use the methodology in [101]. This methodology proves that using autosynchronisation allows the estimated parameters to converge to the true parameters and is thus an ideal estimator for the

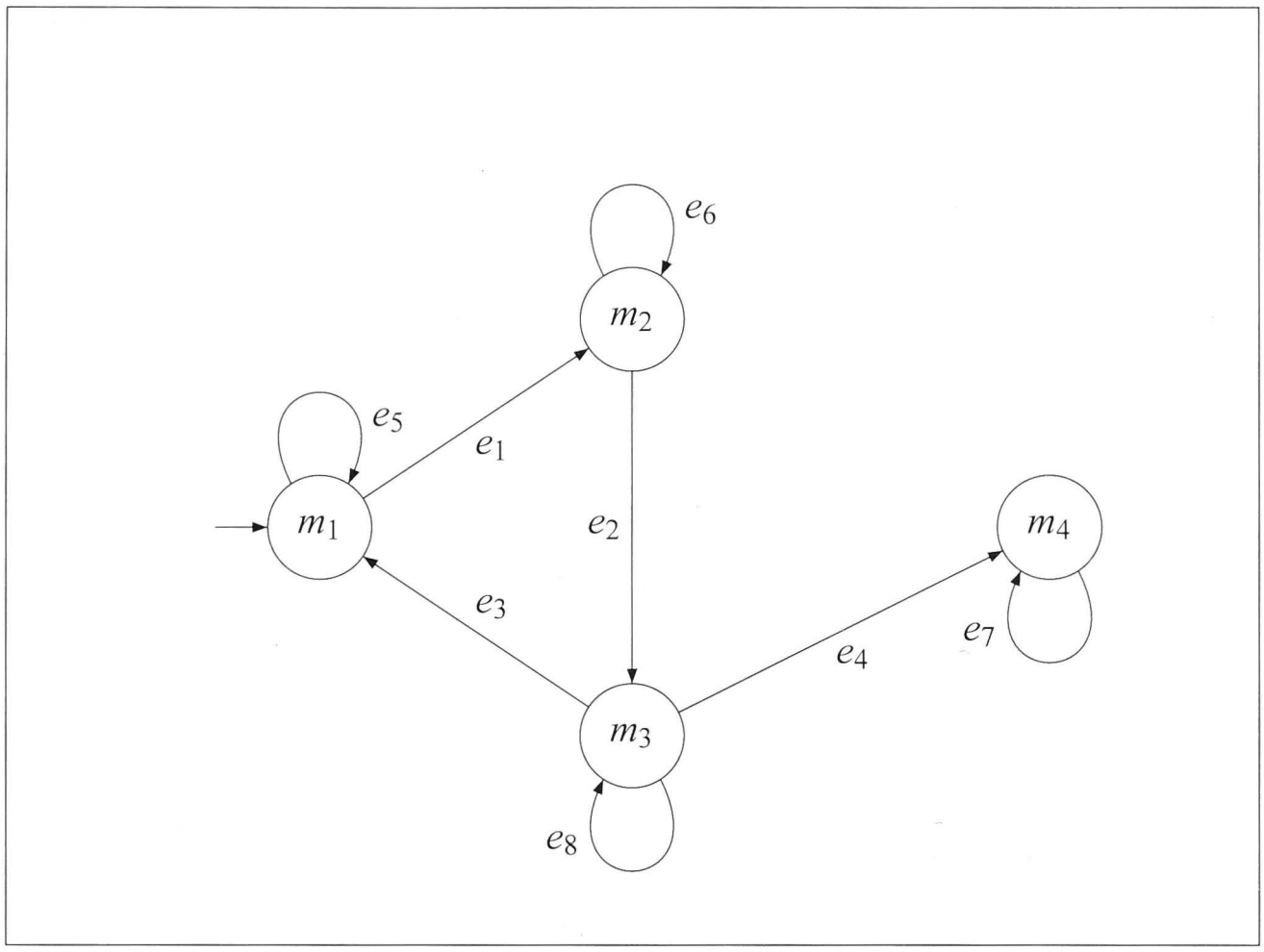


Figure 7.2: The automaton for the discrete operation of the hybrid system in this example.

Mode	Structure	Parameters
1	Rossler	[0.2 0.2 5.7]
2	Rossler	[0.15 0.4 8.5]
3	Lorenz	[10 28 1 0 2.667]
4	Lorenz	[10 50 1 0 20]

Table 7.1: The true dynamical structure and parameters for each of the four operational modes of the hybrid system.

current diagnosis methodology. As modes one and two are both Rossler attractors we need only a single estimator to be able to estimate the parameters for both operational modes. Similarly we need only a single estimator for modes three and four as they are both Lorenz attractors. Using this approach the parameter estimates for both the Rossler (Fig. 7.3) and Lorenz (Fig. 7.4) attractors can be seen.

The parameter estimates for the Rossler attractor are convergent in the period $t \in [0, a]$ where $a \approx 250$. This is the period when the true underlying dynamics are a Rossler attractor and thus the estimation algorithm is able to accurately identify the parameters in this regime. After $t = 250s$ when the true underlying dynamics are represented by a Lorenz attractor the parameter estimates oscillate wildly and are forcibly limited to a magnitude of 50 to prevent excessive divergence of the parameter estimates. Similarly the parameter estimates for the Lorenz attractor are convergent in the period $t \in [a, 400]$ where $a \approx 250$ where the reasons are identical to those presented for the

Transition	Time (s)
e_1	100
e_2	250
e_4	350

Table 7.2: The time and events of the mode changes induced into the simulation that we are attempting to detect using the hybrid systems diagnosis methodology presented in this chapter.

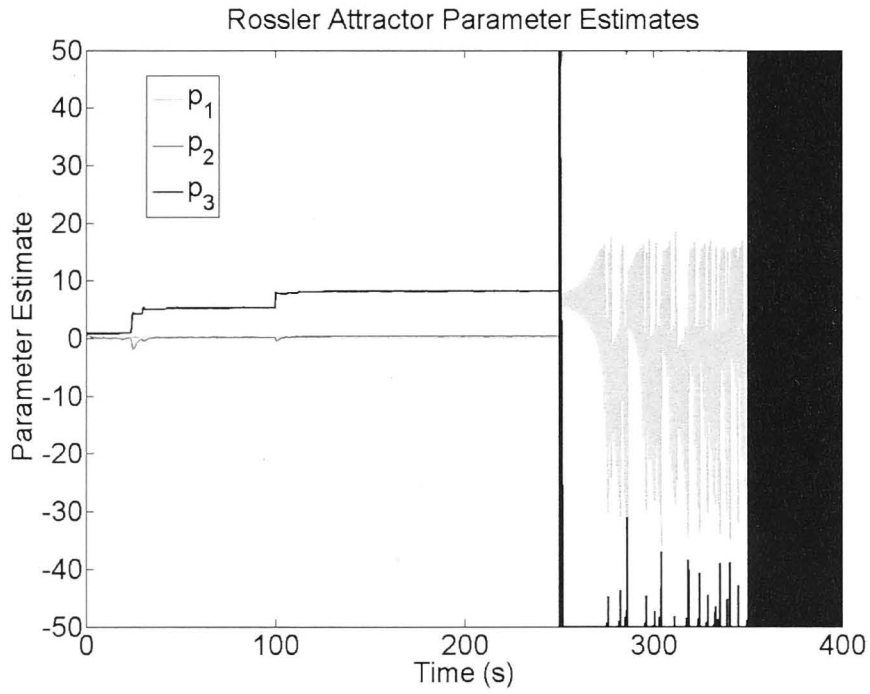


Figure 7.3: Rossler attractor parameter estimates.

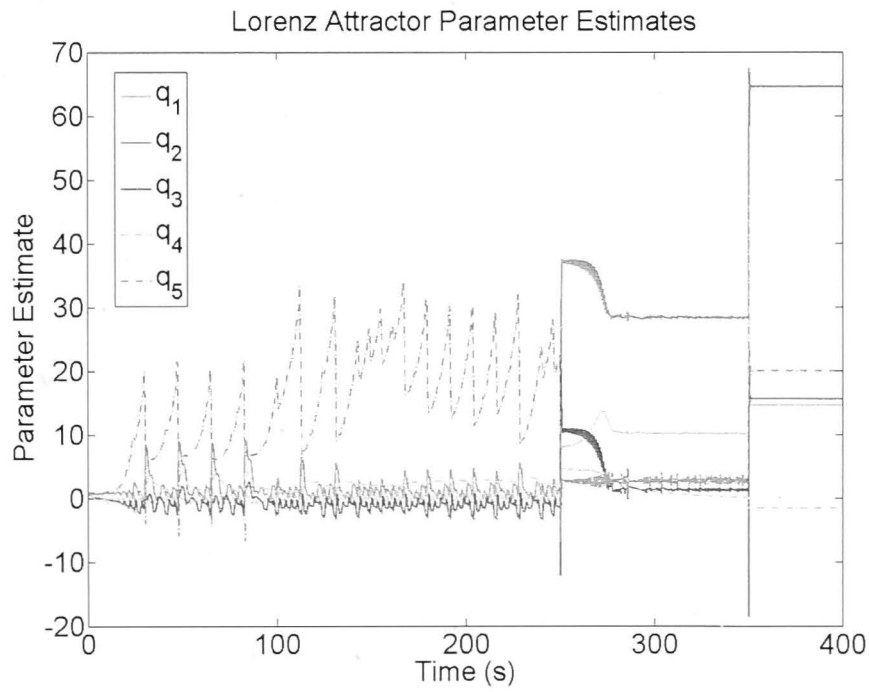


Figure 7.4: Lorenz attractor parameter estimates.

Mode	Structure	Parameters
1	Rossler	[0 0 5]
2	Rossler	[0 0 8]
3	Lorenz	[9 25 0 0 2]
4	Lorenz	[9 48 0 0 18]

Table 7.3: The apriori parameter estimates for each operational mode of the hybrid system. Emphasis must be placed on the fact that there are a multitude of ways of generating potential cost functions that do not require apriori information and thus this approach is descriptive rather than prescriptive.

Rossler attractor. What is clear from these results is that it is possible to accurately estimate the parameter vector when the structure of the dynamics is assumed, thus providing a good justification for the methodology proposed in this chapter.

In order to compute the relative probabilities of each operational mode we define the following cost functions:

$$J_{\text{rossler}} = \frac{1}{3}\sum_{i=1}^{3}(p_i - p_{i_{est}})^2 \tag{7.12}$$

$$J_{\text{lorenz}} = \frac{1}{5}\sum_{i=1}^{5}(q_i - q_{i_{est}})^2 \tag{7.13}$$

These cost functions represent the "average" error of the parameter estimates where $p_{i_{est}}$ and $q_{i_{est}}$ are apriori estimates of the parameters of each system (given in Table 7.3 for this example). This can be seen as a similar approach to residual generation except that residuals are being computed on the parameter estimate rather than the output trajectories. This particular implementation however is by no means the only approach and estimators that do not require any information apriori (as in [16]) can be used equally well.

We use the notation J_i to indicate the i-th cost function where the cost function and the apriori parameter estimates should be clear from Table 7.3 and Eq. (7.8) and (7.11). From this perspective the probabilities of each mode can be defined as

$$P(i|Y(t)) = \frac{1}{3}(1 - \frac{J_i}{\sum_i J_i}) \tag{7.14}$$

where $Y(t) = \{y(0), y(1), \cdots, y(t)\}$ is the discrete observation vector where observations are one second apart. The probabilities for each operational mode are shown in Fig. 7.5. We see that the highest probability accurately reflects the true operational mode, however we are interested in determining if these mode probabilities lead to correct determination of the events that occurred.

At each observation step we determine the transition probabilities using Eq. (7.4). Given the small number of operational modes in the system we pass all $m^2 = 16$ candidate transitions to the discrete diagnosis algorithm to complete the diagnosis. Using the discrete diagnosis algorithm

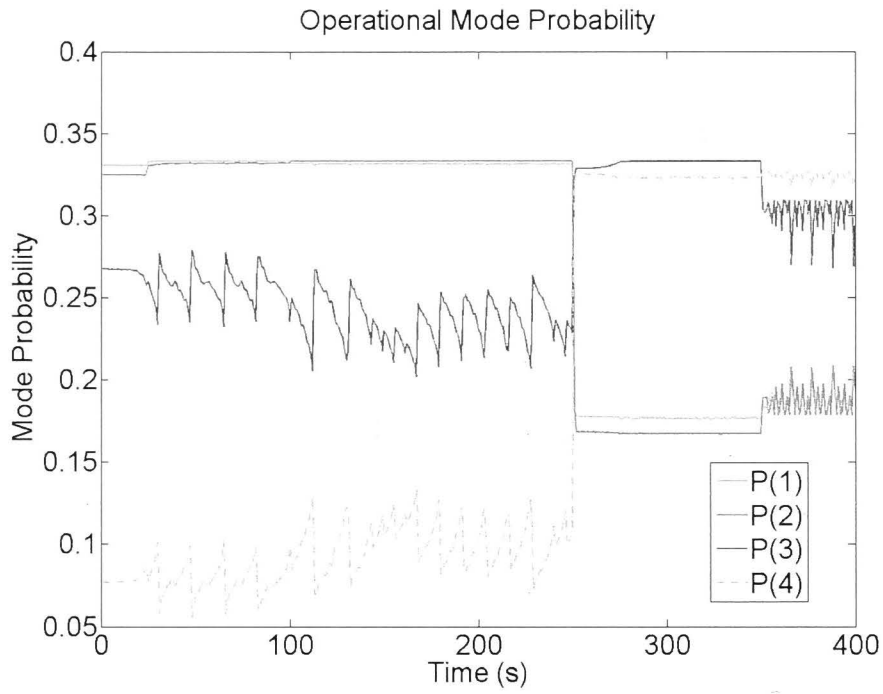


Figure 7.5: Operational mode probabilities for each of the four operating modes after each observation.

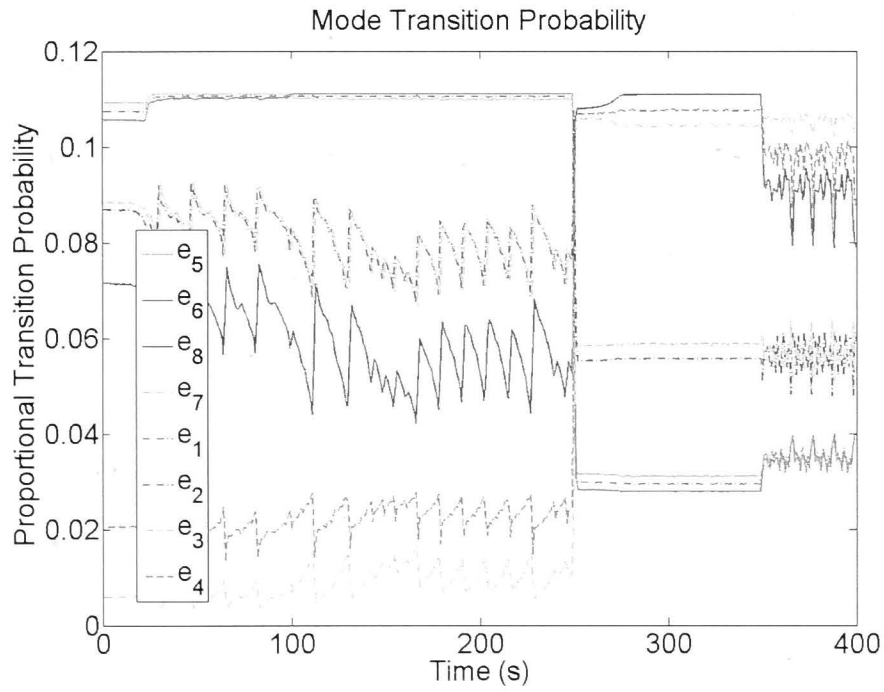


Figure 7.6: Transition probabilities between each of the four operating mode after each observation.

Event	Occurrence Time (s)
e_1	151
e_2	251
e_4	351

Table 7.4: The diagnosed events and the times they occurred using the hybrid systems approach to diagnosis proposed in this chapter.

we are able to determine that only eight of the sixteen mode transitions are actually possible at some point during the time of operation of the system and the transition probabilities for these eight transitions are shown for each observation in Fig. 7.6. The diagnosis of the transitions then follows that the mode transition that is possible (as determined by the discrete diagnoser) with the highest probability (as determined by the estimation and change-point detection methods) is the most likely event that has occurred and is returned as the diagnosis. Using this approach we get the events and the time they occurred in Table 7.4.

We see that the proposed diagnosis methodology is able to accurately determine the occurrence of events in a hybrid system in an online manner. These results highlight that the overlap of approaches from the FDI and DX communities offers potential for powerful diagnosis methods that can be applied to real world systems.

7.5 Hybrid Dynamical Networks

Networks were introduced in chapter 4.4 as a set of interconnected components in the physical sense. They have emerged as a way of describing and analysing large numbers of systems that interact. Understanding the interaction of these systems is of considerable interest in order to better understand the collective operation of the whole. In this part of the thesis, we look at networks not just from the discrete but also from the continuous diagnosis perspective, within a hybrid framework.

We now present a modified definition of a network from the one in chapter 4. It is essentially a similar definition as presented previously but with a focus on the links of the network that will enable us to define dynamical interactions between components.

Definition 50 (Network). A **network** is a directed graph formed by the interconnection of a set of nodes (N) through the set of links (L) by the maps $init(L) : L \to N$ and $ter(L) : L \to N$, that assign to each edge an initial and terminal vertex respectively and will be written as $G = (N, L)$.

We are particularly interested in hybrid dynamical networks. Building on the definition for a hybrid system given in chapter 6, we present additional concepts needed to define hybrid dynamical networks.

Definition 51 (Hybrid Dynamical Node). A hybrid dynamical node N_k, which is part of a network of nodes, is a hybrid system whose operation, in isolation, is governed by a set of hybrid dynamical

equations that describe implicitly how the state of the system changes with time. These equations are essentially the same as equation 6.1 but augmented with a subscript k to denote that the system is the k^{th} node of a network.

$$\begin{aligned} \dot{\mathbf{x}}_{\mathbf{k}} &= f_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}, \mathbf{u}_{\mathbf{k}_{\tau_{\mathbf{k}}}}, \theta_{k_{\tau_k}}) \\ \mathbf{y}_{\mathbf{k}} &= h_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}) \end{aligned} \tag{7.15}$$

where $\dot{\mathbf{x}}_{\mathbf{k}}$ is the time derivative of the node state vector $\mathbf{x}_{\mathbf{k}} \in \mathbb{R}^n$ that contains the quantities needed to describe the operating condition of the node N_k. $\mathbf{u}_{\mathbf{k}_{\tau_{\mathbf{k}}}} \in \mathbb{R}^m$ is the local control vector that gives the values of the local control inputs used to control the operation of the dynamical node k. $\mathbf{y}_{\mathbf{k}} \in \mathbb{R}^p$ is the local output that gives the values of the measurable output of the dynamical node. $\theta_{k_{\tau_k}}$ represents the set of system parameters for node k. $f_{k_{\tau_k}}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ represents the internal and control functions, which together with system parameters $\theta_{k_{\tau_k}}$ describes the dynamical evolution of node N_k. $h_{k_{\tau_k}}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^p$ is the output function.

Definition 52 (Interconnection)**.** An interconnection (L_{kj}) is a directed physical or information theoretic link between two dynamical nodes (N_k) and (N_j), where $init(L_{kj}) : L_{kj} \to N_k$ and $ter(L_{kj}) : L_{kj} \to N_j$. We define the value of an interconnection between two dynamical nodes (N_k) and (N_j) as $(l_{kj}(\mathbf{x}_{\mathbf{k}}, \mathbf{y}_{\mathbf{j}}) : \mathbb{R}^{n_k} \times \mathbb{R}^{p_j} \to \mathbb{R}^{n_k})$

We now define a hybrid dynamical network which is essentially a network of interconnected hybrid systems.

Definition 53 (Hybrid Dynamical Network)**.** A hybrid dynamical network $G = (N, L)$ is a network where the set of nodes (N) and the set of interconnections (L) are as given previously (cf. Def 51 and Def 52 respectively). Mathematically a hybrid dynamical network consists of nodes (N_k) with a set of internal states $(\mathbf{x}_{\mathbf{k}} \in \mathbb{R}^{n_k})$, a set of local control inputs $(\mathbf{u}_{\mathbf{k}_{\tau_{\mathbf{k}}}} \in \mathbb{R}^{m_{k_{\tau_k}}})$, a set of measurable outputs given by $(\mathbf{y}_{\mathbf{k}} \in \mathbb{R}^{p_k})$ and a set of system parameters $(\theta_{k_{\tau_k}} \in \mathbb{R}^{q_{k_{\tau_k}}})$. We also have an associated set of internal and control dynamics $(f_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}) : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k})$ and output function given by $(h_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}) : \mathbb{R}^{n_k} \to \mathbb{R}^{p_k})$. Furthermore each dynamic node is connected to other dynamic nodes through non-linear interconnections $(l_{kj_{\tau_k}}(\mathbf{x}_{\mathbf{k}}, \mathbf{y}_{\mathbf{j}}) : \mathbb{R}^{n_k} \times \mathbb{R}^{p_j} \to \mathbb{R}^{n_k})$. $f_{k_{\lambda_k}}$, $g_{k_{\lambda_k}}$ and $h_{k_{\lambda_k}}$ are continuous functions. This gives the dynamical model of the k-th dynamical system in a dynamical network as:

$$\begin{aligned} \dot{\mathbf{x}}_{\mathbf{k}} &= f_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}, \mathbf{u}_{\mathbf{k}_{\tau_{\mathbf{k}}}}, \theta_{k_{\tau_k}}) + \sum_j l_{kj_{\tau_k}}(\mathbf{x}_{\mathbf{k}}, \mathbf{y}_{\mathbf{j}}) \\ \mathbf{y}_{\mathbf{k}} &= h_{k_{\tau_k}}(\mathbf{x}_{\mathbf{k}}) \end{aligned} \tag{7.16}$$

for $k \in 1, 2, \cdots, K$. The switching signal Ψ_k (as defined in definition 48) takes values in $M_k = \{1, 2, \cdots, m_k\}$, where M_k is the set of operating modes of the system.

Hybrid dynamical networks allow the continuous time operation of a single node or link to be broken into discrete operating modes. An operating mode of a hybrid system is a distinct type of

behaviour which is governed by a set of dynamics that is different from other modes of the system. This provides an elegant way of abstracting the continuous time dynamics of the network into a form that allows faults and other changes in the network dynamics to be accurately modelled.

The switching sequence Ψ_k describes the sequence of mode changes of the k-th hybrid system with the evolution of time:

$$\Psi_k = \{(j_{k_0}, t_{k_0}), (j_{k_1}, t_{k_1}), \cdots, (j_{k_z}, t_{k_z}) | j_{k_z} \in M_k, z \in \mathbb{Z}\}$$

where x_{k_0}, t_{k_0} are the initial system state and time respectively of the k-th system and $\mathbb{Z}_k$ is the set of nonnegative integers. We note that $t_1 = t_2 = \cdots = t_K = t$, that is all systems have the same time. When $t_k \in [t_{k_z}, t_{k_{z+1}})$, $\lambda_k = j_{k_z}$, we say the j_{k_z}-th mode of system k is active and the trajectory of the switched system $(\mathbf{x_k}(t))$ is defined as the trajectory of the system $(\mathbf{x_{k_{j_k}}}(t))$ in mode $j_k \in M_k$.

When there is only a single system in the network we recover the usual definition for a single hybrid dynamical system. We assume that the state of the system is continuous and thus does not exhibit abrupt changes at the instant of switching. The entire internal, control and network dynamics of the network at time t can be characterised by a K-tuple $([\tau_1(t), \tau_2(t), \cdots, \tau_K(t)] \in M_1 \times M_2 \times \cdots \times M_K)$.

7.5.1 Hybrid Dynamical Modes

We present here some definitions and concepts relevant to the representation of hybrid dynamical nodes as graphs in order to more clearly define what we mean by the modes of a hybrid dynamical network.

Definition 54 (Set of States)**.** The set of states (X) of a dynamical node is a discrete set composed of the functions ($x_i : \mathbb{R}_+ \to \mathbb{R}$) that for each $t \geq 0$ returns the value of the node state defined by the function x_i.

Definition 55 (Set of Outputs)**.** The set of outputs (Y) of a dynamical node is a discrete set composed of the functions ($y_i : \mathbb{R}_+ \to \mathbb{R}$) that for each $t \geq 0$ returns the value of the node output defined by the function y_i.

Definition 56 (Set of Controls)**.** The set of controls (U) of a dynamical node is a discrete set composed of the functions ($u_i : \mathbb{R}_+ \to \mathbb{R}$) that for each $t \geq 0$ returns the value of the node controls defined by the function u_i.

We now define our dynamical node as a graph progressing in a similar but slightly more general way than was presented in [17].

Definition 57 (Dynamical Node Graph)**.** A dynamical node graph is a directed graph representing the internal interconnection of the set of states, controls and network inputs and outputs of a dynamical node ($N = X \cup U \cup Y$) through the set of links ($L = L_{XX} \cup L_{UX} \cup L_{YX} \cup L_{XY}$) by the maps $init(L) : L \to N$ and $ter(L) : L \to N$, that assign to each edge an initial and terminal vertex respectively.

The set of links defined above is formally defined as:

Definition 58 (Set of Links)**.** The internal interconnection links between states for the k-th dynamical node is given by $L_{XX_k} = \{(x_{k_i}, x_{k_j}) | \frac{\partial x_{k_j}}{\partial x_{k_i}} \neq 0\}, \quad k_i, k_j \in 1, \cdots, n_k$. The internal interconnection links between control inputs and states for the k-th dynamical node is given by $L_{UX_k} = \{(u_{k_i}, x_{k_j}) | \frac{\partial x_{k_j}}{\partial u_{k_i}} \neq 0\}, \quad k_i \in 1, \cdots, m_k, \quad k_j \in 1, \cdots, n_k$. The internal interconnection links between states and outputs for the k-th dynamical node is given by: $L_{XY_k} = \{(x_{k_i}, y_{k_j}) | \frac{\partial y_{k_j}}{\partial x_{k_i}} \neq 0\}, \quad k_i \in 1, \cdots, n_k, \quad k_j \in 1, \cdots, p_k$ and the interconnection links between all the network outputs and internal states for the k-th dynamical node is given by $L_{YX_k} = \{(y_{k_K}, x_{k_j}) | \frac{\partial x_{k_j}}{\partial y_{k_K}} \neq 0\}, \quad k_K \in 1, \cdots, \sum_k m_k, \quad k_j \in 1, \cdots, n_k$ where (v_1, v_2) denotes a directed edge from node $v_1 \in N$ to node $v_2 \in N$.

We also define the capacity of a structural dynamic graph as a measure of the magnitude of each of the non-zero links defined in the link sets above.

Definition 59 (Capacity of a Dynamical Node Graph)**.** The capacity of a structural dynamic graph is a set C_L that assigns to each link $L_i \in L$ a measure of the capacity of that link. For the link sets defined in Def. 58 we have the following capacity sets for the k-th dynamical node: $C_{L_{XX_k}} = \{\frac{\partial x_{k_j}}{\partial x_{k_i}} | (x_{k_i}, x_{k_j}) \in L_{XX_k}\}$, $C_{L_{UX_k}} = \{\frac{\partial x_{k_i}}{\partial u_{k_j}} | (u_{k_j}, x_{k_i}) \in L_{UX_k}\}$, $C_{L_{XY_k}} = \{\frac{\partial y_{k_i}}{\partial x_{k_j}} | (x_{k_j}, y_{k_i}) \in L_{XY_k}\}$ and $C_{L_{YX_k}} = \{\frac{\partial x_{k_i}}{\partial y_{k_j}} | (y_{k_j}, x_{k_i}) \in L_{YX_k}\}$

From these definitions, we can define the concept of a change in mode next.

7.5.2 Mode Changes

A change in mode is a change of the fundamental governing dynamics and we can see this as follows. For the switching sequence of the k-th node in a hybrid dynamical network, $\Psi_k = \{(j_{k_0}, t_{k_0}), (j_{k_1}, t_{k_1}), \cdots, (j_{k_z}, t_{k_z}) | j_{k_z} \in M_k, z \in \mathbb{Z}\}$ we take $L_{j_{k_z}} = L_{XX_{j_{k_z}}} \cup L_{UX_{j_{k_z}}} \cup L_{YX_{j_{k_z}}} \cup L_{XY_{j_{k_z}}}$ to be the total link set in the time interval $[t_{k_z}, t_{k_{z+1}})$ and $C_{L_{j_{k_z}}} = C_{L_{XX_{j_{k_z}}}} \cup C_{L_{UX_{j_{k_z}}}} \cup C_{L_{YX_{j_{k_z}}}} \cup C_{L_{XY_{j_{k_z}}}}$ to be the equivalent total capacity set in the same interval. A mode change implies that for any $z \in \mathbb{Z}$, $L_{j_{k_z}} \neq L_{j_{k_{z+1}}}$ and/or $C_{L_{j_{k_z}}} \neq C_{L_{j_{k_{z+1}}}}$. That is the change in mode forces the link and/or capacity sets to change, meaning a fundamental change in the fundamental governing dynamics. It should be noted that determining the link and capacity sets is possible using a variety of parameter and system identification methods [69, 23, 58].

7.6 Using Discrete Diagnosis Methods on Hybrid Dynamical Networks

We explain in this section how discrete diagnosis methods can be used on hybrid dynamical networks. We first discuss some of the assumptions that we are making during this analysis.

7.6.1 Analysis Assumptions and Discussion

The assumptions used in this analysis are ideally kept to a minimum to maximise the real-world applications of this approach. Importantly these assumptions do not restrict the method by which a mode changes. It is reasonable to assume that an event may be due to a fault, a change in environmental conditions, the effects of a control choice in a given mode or a variety of other reasons. By looking at the outcome of the event rather than the source of the event we are better placed to use this methodology for dealing with many of the challenges faced in modern dynamical networks. To illustrate, in the example given in chapter 6, section 6.4, lightning (the source) might cause a fault that manifests as overvoltage (the outcome) which can be detected in the governing dynamics of the system.

Our assumptions also allow faults to occur in succession provided they do not occur instantaneously. This is also advantageous as it allows us to deal with cascading type faults which are of significant concern in modern hybrid dynamical networks. The following assumptions are made during the analysis.

7.6.1.1 Mutual Exclusivity of Modes

We assume that the modes (M_k) of a system are mutually exclusive. That is a system (S_k) in the dynamic network can only be in one mode at any time. That is if $\sigma_k = m_i \in M_k$ for $t_k \in [t_{k_i}, t_{k_{i+1}})$ and $\sigma_k = m_j \in M_k$ for $t_k \in [t_{k_i}, t_{k_{i+1}})$ then $m_i = m_j$. This is a natural assumption that ensures any events cause the system to move into one of a discrete set of modes and that one and only one mode is responsible for describing the dynamic evolution of a system in the network at a given time.

7.6.1.2 Timing of Events

We assume that between two observations only one event can occur. The continuous dynamics need to be tracked at regular intervals of time to be able to detect changes. During each of these finite intervals, it is necessary to make the assumption that only one event can occur as this is what the continuous tracking algorithm can deal with. This further implies that for any finite $T > t_0$, there exists a positive integer K_T, which may depend on T, such that during the time interval $[t_0, T]$ each dynamical node (Eq. 7.15) switches no more than K_T times. This ensures that it is practically possible to diagnose a system as the sequence of observations will have information about all the faults that occur in the system.

7.6.1.3 Controller Laws

We are assuming that for a given operational mode it is possible to compute an appropriate controller off-line [61]. The methods for achieving this are too many and varied to enumerate and cover the span of classical and modern control for a period of almost a century. For this reason we

focus purely on the method of choosing an appropriate control strategy when the fault occurs. A simple justification of this can be seen in that if the operating mode of each dynamical node can be chosen independently there are:

$$\prod_k m_k \tag{7.17}$$

K-tuples that characterise the operating condition of the hybrid dynamical network. In networks with large numbers of systems and internal modes this rapidly becomes computationally intractable.

In the majority of dynamical networks a single event or fault will affect a number of systems in the network and thus using diagnosis methodologies allows this number to be drastically reduced so that the off-line computation of controllers is actually feasible. Furthermore if we consider networks where only a subset of the systems are directly observed then without the relationships stored in the diagnosis algorithm it would be impossible to diagnose and control for the effects of those unobserved systems.

7.6.2 Algorithmic Considerations

We outline two methods in which distributed diagnosis can be used in maintaining a fault tolerant network. Algorithm 5 presents a global optimal approach. Algorithm 6 utilises the power of the junction tree technique to address the time criticality issue that often arises in large hybrid dynamical networks to try and avoid cascading failures.

We consider a hybrid dynamical network $S = \{S_1, \ldots, S_K\}$ each of which has an automaton model. We thus have a set of models $Mod_1, \ldots, Mod_K$ available. We have a set $C = \{C_1, \ldots, C_r\}$ of controllers that can be used on the system. At a given time t, we have observations $Obs_1, \ldots, Obs_K$ on the system where each Obs_k is a set of estimates of the mode of a given component S_k at t obtained from parameter estimation and system identification. In Algorithm 5, we pick the

Algorithm 5 Global Optimal Algorithm for diagnosis and control of a hybrid dynamical network

1: At time t_{obs}
2: **input** $\{Mod_1, \ldots, Mod_K\}, \{Obs_1, \ldots, Obs_K\}, C$
3: **for all** $t_{obs} \in t_0, \cdots, t_{final}$ **do**
4: **do** $\Delta_{t_{obs}} := Mod_{t_{obs}} \otimes Obs_{t_{obs}}$
5: Run consistency algorithm (returns globally consistent network)
6: **for all** $t_{obs} \in t_0, \cdots, t_{final}$ **do**
7: pick a control strategy C from C

best control strategy over a globally consistent network. Provided that the diagnosis and global consistency can be performed fast enough, this provides the global optimal solution. However, in a large network, time is often a critical factor in the occurrence of cascading type faults. If we can choose a locally optimal controller after performing local consistency, this could help prevent other dynamical nodes in the system from failing as a potential fault is captured and dealt with as soon as possible. This approach is outlined in Algorithm 6.

Algorithm 6 Optimal Time Critical Algorithm for diagnosis and control of a hybrid dynamical network

1: At time t
2: **input** $\{Mod_1, \ldots, Mod_K\}, \{Obs_1, \ldots, Obs_K\}, C$
3: **for** i=1:K **do**
4: **for all** $t_{obs} \in t_0, \cdots, t_{final}$ **do**
5: **do** $\Delta_i := Mod_i \otimes Obs_i$
6: **while** *not* globally consistent **do**
7: Perform local consistency with neighbours
8: pick a local control strategy C from C

7.6.3 Diagnosis under partial observations

The problem with large networks is that it is not always possible or practical to obtain observations on all dynamical nodes. However, because components share common events, it is still possible to diagnose unobserved nodes by using the diagnosis of the observed ones. For an unobserved component, we take Obs to be the set of all possible modes. We then refine the diagnosis when performing local consistency with other components.

7.7 Demonstrative Example

We will use the fault diagnosis methodology developed to diagnose and recover from faults on a four node complex network (Fig. 7.7). Complex networks (described for example in [68]) have emerged out of the biological and social sciences and are an attempt to explain the behaviour of large interconnected groups. From a dynamical and systems theoretic perspective complex networks consist of independent dynamical nodes connected together. We use the following representation:

$$\dot{\mathbf{x}}_\mathbf{i} = f(\mathbf{x_i}) + c \sum_j a_{ij} \Gamma(\mathbf{x_j} - \mathbf{x_i}) + \epsilon_i(\mathbf{s(t)} - \mathbf{x_i}) \tag{7.18}$$

where it is clear that this complex network is in the dynamical network form presented in Eq. 7.16. We have $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$ is the unique internal dynamics, a_{ij} determines the interconnections in the network and is symmetric. As such $a_{ij} = a_{ji} = 1$ if node i and node j are connected together and zero otherwise. Γ is the feedback connectivity matrix that determines how the difference vector $(\mathbf{x_j} - \mathbf{x_i})$ affects the internal state and c is the scalar gain that determines the magnitude of this effect. We also have the control input $\epsilon_i(\mathbf{s(t)} - \mathbf{x_i})$ where $\mathbf{s(t)}$ is the desired equilibrium state of the dynamical node and the network as a whole and $\epsilon_i \in \{0, \epsilon\}$. That is only some nodes have a local feedback control applied with control gain ϵ. This is referred to in the literature as pinning control [68]. In complex networks the control problem becomes ensuring that the network will synchronise, where synchrony is defined as:

$$\mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_n \to s(t)$$

such that $\dot{\mathbf{x}}(s(t)) = f(s(t))$ is the solution of the isolated nodes.

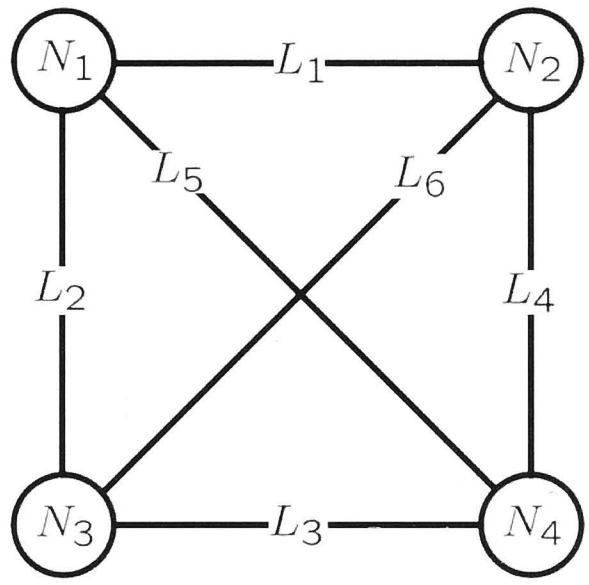


Figure 7.7: The four node, six link complex network analysed in this example.

It is well known in the complex networks community that the ability of the network to synchronise is dependent on the value of c chosen as well as the location and magnitude of the pinning control that is applied to the network. Whilst an arbitrarily large c and pinning control gain (ϵ) can always be chosen this is often physically impossible as well as being inefficient as much more control authority is being commanded than is necessary to achieve the desired synchrony.

Dyamical Model

In our example we choose $f(x_i) = -x_i^2 + 5x_i$, $\Gamma = I$, the four dimensional identity matrix and the network is globally connected. With $c = 1.5$ we can see the resultant state trajectory in Fig. 7.9, where synchrony is rapidly achieved.

We consider faults that affect the structure of the network (similar to [94]). That is we consider a network where the value of $a_{ij} = a_{ji}$ changes when a fault occurs, thus representing the hybrid component of the dynamical network presented in Eq. 7.18. This further implies that only the link sets change when a fault occurs ($L_{i_{k_z}} \neq L_{i_{k_{z+1}}}$). That is the fault causes a fundamental change in the network dynamical structure. The capacity sets do not change($C_{L_{i_{k_z}}} \cap C_{L_{i_{k_{z+1}}}} \subseteq C_{L_{i_{k_{z+1}}}}$) and thus networks of this type are said to be capacity invariant. Even in this simple network the possible structural dynamical fault modes are substantial. We consider ten possible structural faults, four due to node removal and six due to link removal. In the case of node removal we assume that all the links between the removed node and the network are also removed simultaneously. These fault modes correspond to the typical faults found in complex network operation.

Automaton Model

An automaton is shown in figure 7.8 representing mode changes for one node of the network (node N_1). Only structural changes are considered.

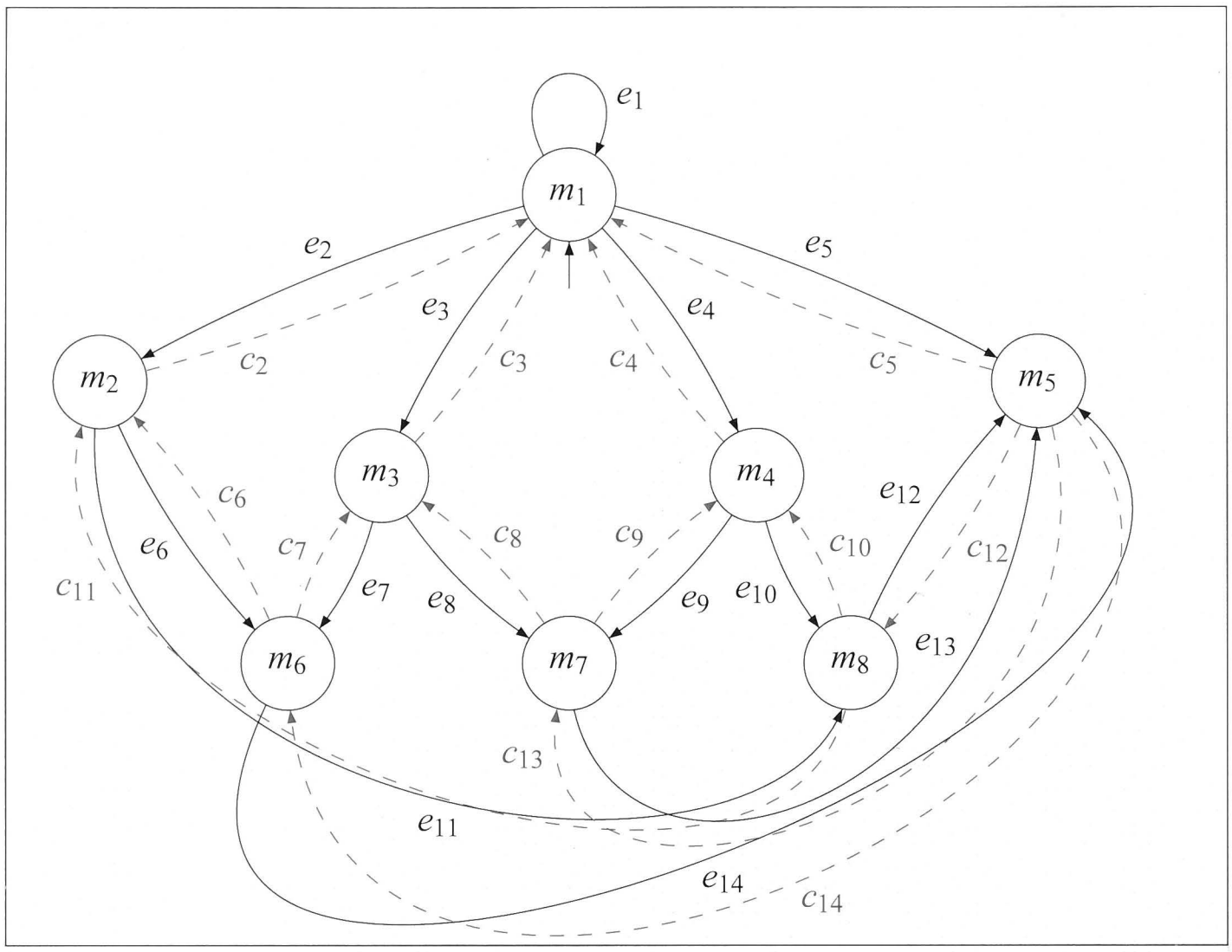


Figure 7.8: Automaton on node N_1 of example network denoting structural mode changes as events

Modes on the automaton correspond to the following:

- m_1: all neighbours observed
- m_2: N_2 and N_4 observed
- m_3: N_2 and N_3 observed
- m_4: N_3 and N_4 observed
- m_5: no neighbours observed
- m_6: N_2 observed
- m_7: N_3 observed
- m_8: N_4 observed

Due to the structure of the network, diagnosis using local observations on node N_1 cannot determine losses of the links L_3, L_4 and L_6. This means that any or all links from the set $\{L_3, L_4, L_6\}$ could be lost without it being possible to detect using observations on node N_1 only. We call the set $\{L_3, L_4, L_6\}$ the blind spot for diagnosis on node N_1. The corresponding events (to the

above modes) are either that nothing happened (denoted by $\emptyset$), or the loss of one of the links or nodes listed below. We note that only one link or node can be lost during a transition due to the assumptions made on the timing of events on the system.

- e_1: $\emptyset, L_3, L_4, L_6$
- e_2: L_2, N_3
- e_3: L_5, N_4
- e_4: L_1, N_2
- e_5: N_1
- e_6: L_5, N_4
- e_7: L_2, N_3
- e_8: L_1, N_2
- e_9: L_5, N_4
- e_{10}: L_2, N_3
- e_{11}: L_1, N_2
- e_{12}: L_5, N_4, N_1
- e_{13}: L_2, N_3, N_1
- e_{14}: L_1, N_2, N_1

Control events are shown in red and are events that could be used to recover from the different faults. What these events are or how they are applied is beyond the scope of this thesis.

Similar automata are built for the other nodes of the network.

Fault Scenarios

We consider two possible fault scenarios in this example:

- The network link L_5 becomes disconnected.
- The network system N_4 becomes disconnected.

We assume that we only have partial observations on the network and thus we are only able to monitor the modes of systems N_1 and N_2. The two faults outlined are important because it is only when we use the diagnosis algorithm, and the observations from both nodes N_1 and N_2, that we are able to exactly determine the fault that occurred and implement an appropriate control strategy.

While we can always assume that we can increase the control effect on the nodes in the case of a fault, this is an inappropriate action as it is known that when network link L_5 becomes disconnected no additional control action is required to ensure network synchrony. Conversely, when network system N_4 becomes disconnected we must apply local control to system N_1 to ensure the network achieves the desired synchrony. In the latter case we apply the local controller described previously using a controller gain of $\epsilon_1 = 2.5$ about the equilibrium point $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = \mathbf{x}_4 = 0$.

The dynamics are simulated in MATLAB and the system is started each time at the states ($[x_1, x_2, x_3, x_4] = [-6.5, -3, -4.95, 6]$) and the fault occurs at $t = 0.1s$. We make observations of the system every $\Delta t_{observations} = 0.4s$. We assume that the internal dynamics of the system are well characterised and thus we are able to measure the network dynamics in isolation. We assume that any inaccuracies in our measurement of the network dynamics can be modelled as zero mean, i.i.d. gaussian noise (η). On this basis our measurement model is:

$$y_{i_{obs}} = c \sum_j a_{ij} \Gamma(\mathbf{x_i} - \mathbf{x_j}) + \eta \tag{7.19}$$

To determine the dynamical structure of the system we are interested in determining all the values of a_{ij}. This is a special case of regression known as sparse regression, more details of which can be found in [15]. We use the algorithm from [15] to determine the values of all the a_{ij}, thus allowing us to determine the link set and thus the modes of the dynamical nodes N_1 and N_2. We re-initialise the algorithm after every observation to ensure we can determine when the dynamical structure of a node changes, thus allowing rapid determination of fault occurrence in the hybrid dynamical network.

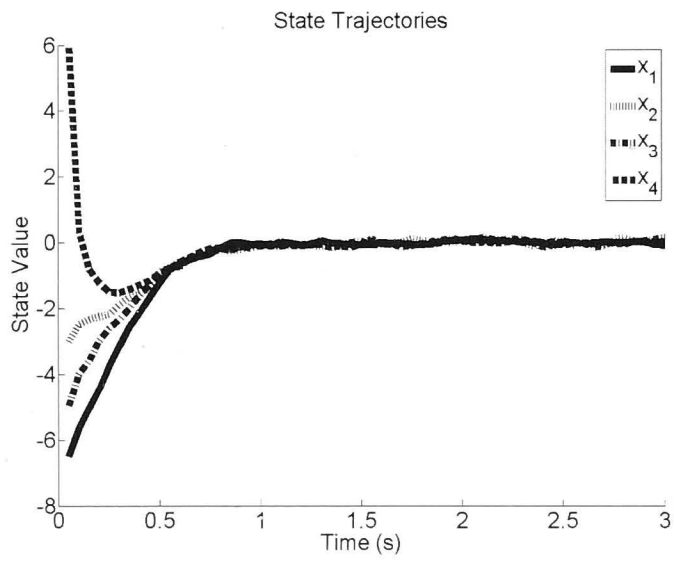


Figure 7.9: The state trajectories of each system in the network when the network is globally connected.

We always assume the initial mode to be fault free and when we induce the first of our faults and the network link L_5 becomes disconnected the diagnosis algorithm is capable of determining that this is the fault that has occurred by synchronising the observations of system N_1 and system N_2. We observe in the resultant state trajectories (Fig. 7.10) that the systems take longer to

achieve synchrony but are still able to do so. Although the performance of the network is reduced (in terms of achieving synchrony) there is no need to inject additional control as the objective is still achieved in a reasonable time as compared to the normal network operation (Fig. 7.9).

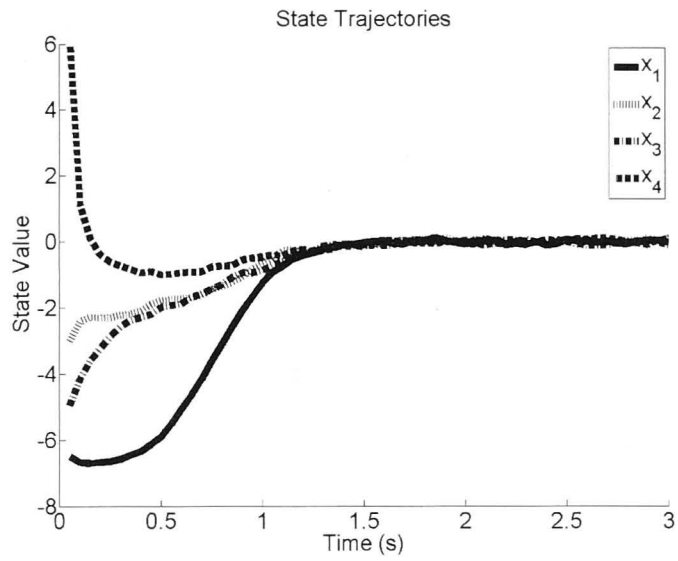


Figure 7.10: The state trajectories of each system in the network when the network link L_5 becomes disconnected due to a fault. The diagnosis algorithm is used here to determine the fault that has occurred and that no additional control input is required to achieve network synchrony.

When the second of the faults is introduced and the network system N_4 becomes disconnected we can see the resultant trajectory that would occur in the absence of diagnosis and thus external control being applied (Fig. 7.11). Alternatively when we use the diagnosis approach, the algorithm again accurately determines the fault that has occurred through the observations of the two nodes N_1 and N_2. In this case the network begins to move rapidly away from equilibrium but is arrested by a local controller that has been switched in when the diagnosis algorithm determines that the node N_4 has been completely disconnected from the network, removing a major stabilising component of the network.

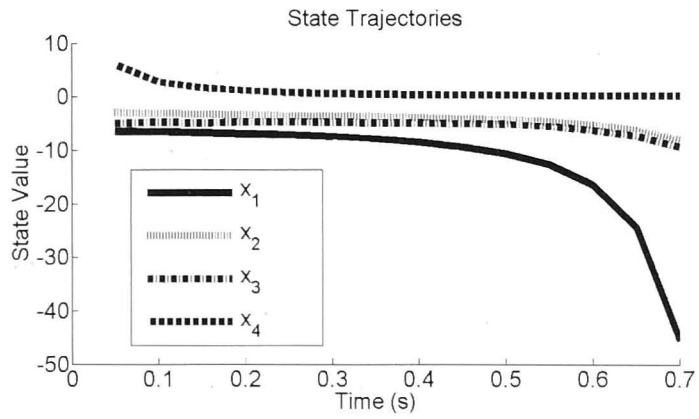


Figure 7.11: The state trajectories of each system in the network when the system N_4 becomes disconnected from the network due to a fault. The diagnosis algorithm is not running and we do not initiate any external control.

We can take a number of key insights from this example. Firstly the importance of being able to accurately and rapidly determine dynamical structure. If this were not possible then it is

unlikely we would be able to accurately determine the fault in such a short period of time and switch in appropriate control strategies to ensure synchrony is achieved. Secondly, although this may appear like a simple example there are many possible faults that can occur in the network and it is only by having a diagnosis algorithm that we are able to accurately identify and recover from a chosen fault. Although in both cases the fault that occurs is related to system N_4 through observing and controlling system N_1 and system N_2 we are able to achieve the desired results. This is a major accomplishment as it removes the necessity for local observations and control at every node in the network and still allows the network performance objectives to be achieved. The optimal location of observation and control is still an area of active research.

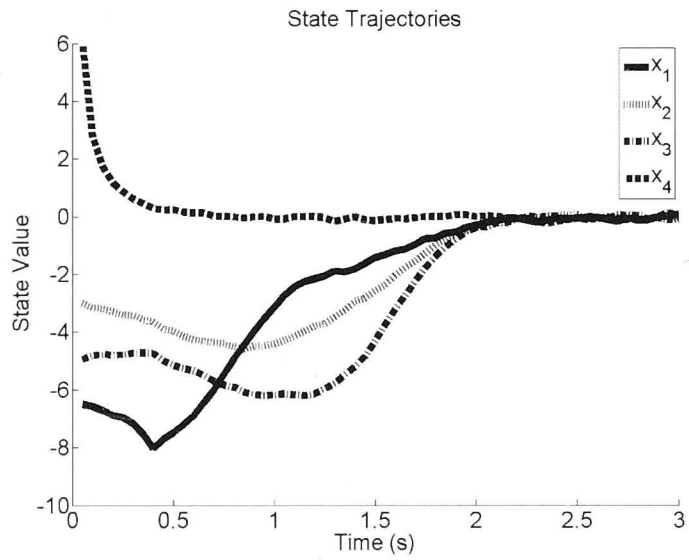


Figure 7.12: The state trajectories when the node N_4 becomes disconnected from the network due to a fault. The diagnosis algorithm detects the fault and switches in a local control input on node N_1 to ensure network synchrony.

7.8 Conclusion

In this chapter we have shown a hybrid approach to doing diagnosis. Techniques from the FDI and DX communities are unified to provide a diagnosis utilising both the continuous and discrete aspects of a system. Changes are detected in the underlying continuous dynamics of the system before being confirmed and refined using discrete diagnosis methods. The hybrid method is illustrated on a hybrid system with four modes and on a hybrid network with four nodes, both exhibiting complex behaviour in their dynamics. The examples presented highlights the capability of this method to accurately diagnose hybrid systems and networks. Furthermore, we have shown how a distributed approach to diagnosis can be used on networks for localised control. This can potentially help avoid cascading failures in the network. This chapter is intended to lay the ground work for the development of advanced algorithms capable of being used in real-world scenarios.

One area of future work is to extend diagnosis to other types of faults than just structural faults on a network. This will add an additional layer of complexity to the methodology but is an important extension for applicability to real world scenarios. Another area of future work is

testing on a variety of simulated and real world systems to understand better how best to adapt this approach in providing a robust hybrid systems diagnoser for real world applications.

Chapter 8

Diagnosability of Networks of Hybrid Systems

8.1 Introduction

Chapter 7 presented a framework to represent hybrid dynamical systems and networks and showed how it could be used to perform diagnosis. The interconnection of many dynamical systems is known to lead to complex, emergent behaviour not observed in the individual dynamic systems. The accurate diagnosis, that is the detection and isolation, of faults and other events in dynamical networks has thus emerged as an important problem in characterising the operation of such networks. However, before attempting diagnosis on a system, it is important to know whether it is possible to identify all know faults that might occur on the system. In other words, we need to ensure the *diagnosability*, as defined in definition 19, of the system. Diagnosability can be thought of as the abstract determination of whether the correct, minimal diagnosis can be made and is significantly complicated in dynamical networks due to the interaction of the fault and event characteristics of the many interconnected systems.

We present here a general characterisation of the diagnosability conditions for the detection and isolation of multiple, arbitrary sequential or simultaneous events in hybrid dynamical networks. We give the conditions for which arbitrary events (including faults) in hybrid networks can be detected and isolated using a general class of indicator functions. These results emerge from the overlap of the control theoretic fault detection and isolation and diagnosis communities. The interaction of the many systems in the network is exploited to achieve diagnosability conditions dependent only on the number of events in the network rather than the number of interconnected systems. The algorithmic complexity of the diagnosability conditions and methods of choosing a minimal indicator set that guarantee diagnosability are also addressed.

8.2 Hybrid Dynamical Networks

Dynamical networks emerge from the interconnection of many individual dynamical systems, where each dynamical system is a node in the network. Hybrid dynamical networks are covered in chapter 7. We are interested in the definition for a hybrid dynamical network as given in definition 53. We reproduce the equation describing the dynamical model of the k-th dynamical system in a dynamical network here. The governing dynamics for each node (N_k) of the hybrid dynamical network as given in equation 7.16 in chapter 7 and reproduced here:

$$\dot{\mathbf{x}}_\mathbf{k} = f_{k_{\tau_k}}(\mathbf{x}_\mathbf{k}, \mathbf{u}_{\mathbf{k}_{\tau_\mathbf{k}}}, \boldsymbol{\theta}_{k_{\tau_k}}) + \sum_j l_{kj_{\tau_k}}(\mathbf{x}_\mathbf{k}, \mathbf{y}_\mathbf{j})$$

$$\mathbf{y}_\mathbf{k} = h_{k_{\tau_k}}(\mathbf{x}_\mathbf{k})$$

for $k \in 1, 2, \cdots, K$. The switching signal Ψ_k (as defined in definition 48) takes values in $M_k = \{1, 2, \cdots, m_k\}$, where M_k is the set of operating modes of the system.

8.3 Running Example

We will use an example of a hybrid dynamical network throughout this chapter that has been introduced in section 7.6 of chapter 7 . The network is a simple four node, fully interconnected hybrid dynamical network as seen in figure 7.7 and reproduced for ease of reading this chapter. We are interested in determining the structural diagnosability of the network. The structural diagnosability refers to the ability to accurately detect node and/or link failures in this network.

A link failure occurs when a given link between two nodes fails and a node failure occurs when a node, and all its interconnections, are simultaneously removed from the network. These structural changes can be well codified in the theory of hybrid dynamical networks outlined earlier where a change in the interconnection structure of the k-th node corresponds to a different operational mode (τ_k) and thus different interconnection dynamics ($l_{kj_{\tau_k}}(\mathbf{x}_\mathbf{k}, \mathbf{y}_\mathbf{j})$).

From this perspective there are ten distinct faults that can occur, corresponding to the four node failures and six link failures. A link failure between two nodes, caused by either a link or node failing, can be seen to impact both nodes to which it is connected. Hence, it is possible to determine the occurrence of the link failure from observing either of the connected nodes. It is of interest to determine the conditions under which these node and link failures are diagnosable. The example will be continued in the *Example Continued* sections throughout the chapter.

8.4 Modes and Events

The discrete operating modes of each system in a hybrid dynamical network completely describe the operation of the network as a whole. These modes can correspond to arbitrary operating conditions, some nominal and some induced through the occurrence of a fault or other event in the

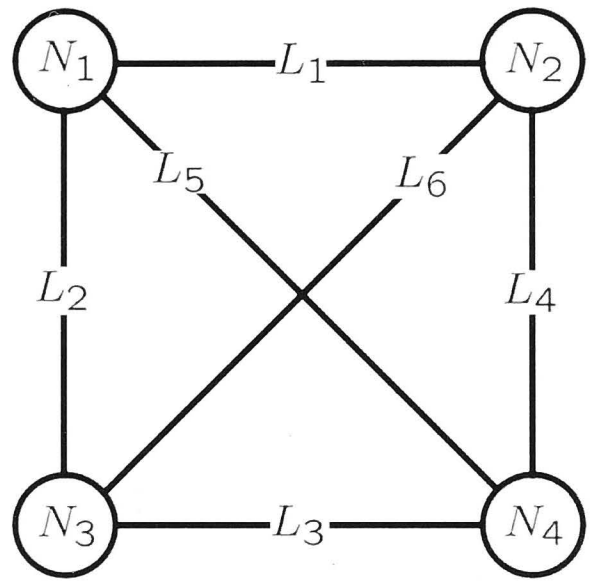


Figure 8.1: Figure 7.7 showing the four node, six link complex network example introduced in the previous chapter.

network. We use the term events to be very general and include faults and other events of interest that can be characterised by a change in operating mode. We assume that there are a finite number of events (including faults) that can occur within the network and we can thus define the set (E) of n network events:

$$e_i \in E, \ \forall i \in \{1, 2, \cdots, n\} \tag{8.1}$$

representing all the possible events that can occur in the network. We then define the status of an event (e_i) as:

$$e_{i[t_\zeta, t_{\zeta+1})} = \begin{cases} 0 & \text{Event } e_i \text{ hasn't occurred in } [t_\zeta, t_{\zeta+1}). \\ 1 & \text{Event } e_i \text{ has occurred in } [t_\zeta, t_{\zeta+1}). \end{cases} \tag{8.2}$$

This representation, as we will see later, has important benefits for computing the diagnosability of the hybrid dynamical network. Additionally $[t_\zeta, t_{\zeta+1})$ is a sampling interval for arbitrary t_ζ and $t_{\zeta+1}$ where $t_{\zeta+1} > t_\zeta$. We will see later that this sampling interval is necessary for the indicator functions of Section 8.5 to be robustly defined. There is an explicit assumption here that an event e_i can only occur once in the interval $[t_\zeta, t_{\zeta+1})$. The sampling interval can be varied depending on the requirements of the particular network and the sampling intervals cover the entire operational time of the network thus:

$$\bigcup_{\zeta \in \mathbb{Z}} [t_\zeta, t_{\zeta+1}) = [t_0, t_\infty) \tag{8.3}$$

where $\mathbb{Z}$ is the set of nonnegative integers.

We also define the set of active events $\Omega_{[t_\zeta, t_{\zeta+1})} \subseteq E$ to be the set of events that are occurring in the time interval $[t_\zeta, t_{\zeta+1})$:

$$\Omega_{[t_\zeta, t_{\zeta+1})} = \{e_i | e_{i[t_\zeta, t_{\zeta+1})} = 1\} \tag{8.4}$$

It should be noted that the true value of $e_{i[t_\zeta, t_{\zeta+1})}$ and $\Omega_{[t_\zeta, t_{\zeta+1})}$ is never known explicitly and this is

what we seek to estimate with a given diagnosis.

The events that occur in the network cause a corresponding mode change in some or all of the hybrid systems in the network. It is common to represent the relationship between the network events and the discrete operating modes of the k-th system by a deterministic finite automaton as in [20]. This gives:

Definition 60 (Deterministic Finite Automaton (DFA))**.** A deterministic finite automaton for the discrete behavior of the k-th hybrid system in the network is given by the tuple:

$$S_{k_{disc}} = \langle M_k, \bar{E}_k, \mathcal{T}_k \rangle \tag{8.5}$$

where for the k-th system M_k is the finite set of system modes, $\bar{E}_k \subseteq E$ is the set of events which cause a mode transition and $\mathcal{T}_k$ is the transition function that maps an event and mode into a new mode, $\mathcal{T}_k : \bar{E}_k \times M_k \rightarrow M_k$.

The DFA formalizes the relationship between the operating modes of each of the hybrid systems and the events that occur within the network. In large interconnected networks it is common that a single event will appear in the DFA for a number of systems in the network. This corresponds to a single event being responsible for mode changes in more than one hybrid system in the network. We can exploit this behavior to determine the operating mode of each system in the network without needing to determine the operating mode of each system explicitly. This can be shown by:

Lemma 2 (Hybrid System Mode Determination)**.** Given the automaton ($S_{k_{disc}}$) and the initial operational mode ($\tau_k(t_0)$) of the k-th system and assuming that no more than one event in $\bar{E}_k$ occurs in each sampling interval $[t_\zeta, t_{\zeta+1})$ then knowledge of $\Omega_{[t_\zeta, t_{\zeta+1})}$ (where $\Omega_{[t_\zeta, t_{\zeta+1})} \cap \bar{E}_k$ contains the relevant events for the k-th system) for all sampling intervals $[t_\zeta, t_{\zeta+1})$ will allow determination of $\tau_k(t_{\zeta+1})$ for all $t_{\zeta+1}$.

Proof. Given $S_{k_{disc}}$ and $\tau_k(t_\zeta)$ we can determine $\tau_k(t_{\zeta+1})$ after the interval $[t_\zeta, t_{\zeta+1})$ using:

$$\tau_k(t_{\zeta+1}) = \mathcal{T}_k(\tau_k(t_\zeta), \Omega_{[t_\zeta, t_{\zeta+1})} \cap \bar{E}_k) \tag{8.6}$$

□

Remark 8.4.1. Although we require that only a single event occurs in a given sampling interval for each system, this does not restrict the possibility that multiple events can occur simultaneously as multiple events affecting different systems in the network may occur simultaneously. Additionally, assuming that the sampling interval is sufficiently small compared to the fault occurrence interval then this assumption can be seen to be very unrestrictive.

From this we see that if every event in the network can be determined in each time interval then the K-tuple $([\tau_1(t_{\zeta+1}), \tau_2(t_{\zeta+1}), \cdots, \tau_K(t_{\zeta+1})])$ can be reliably determined for all $t_{\zeta+1}$. For this

reason the diagnosability of the operational mode of each system in the network can be reduced to the diagnosability of all the events in the network. Finding reliable methods of determining the events that are occurring in the network is the topic of the following section.

8.5 Indicator Functions and Event Detection

It is common in dynamical systems to have indicators; quantities that indicate the transition of a system between modes or that measure the occurrence of an event directly. We saw in Section 8.4 that if it is possible to determine the occurrence of all events in the hybrid network then it is possible to know the operational mode of all systems in the network. In showing this we have effectively decoupled the problem of detecting and isolating network events and the determination of the operational mode of every system in the network which is the eventual goal. We can formalize the concept of indicator functions as follows.
We define the set (S) of m indicator functions as:

$$s_j \in S, \ \forall j \in \{1, 2, \cdots, m\}. \tag{8.7}$$

where:

$$s_j := \langle g_j, \Phi_j, E_{s_j}, \nu_j \rangle \tag{8.8}$$

is a tuple where $\Phi_j \subseteq Q_j \subseteq \mathbb{R}^{\kappa_j}$ and Φ_j and Q_j are well-defined. The indicator function $g_j : \mathbb{R}^{q_j} \rightarrow Q_j \subseteq \mathbb{R}^{\kappa_j}$ is defined as:

$$g_j(Y_{j_{[t_\zeta, t_{\zeta+1})}}) \in \begin{cases} \Phi_j \subseteq Q_j & \text{if } E_{s_j} \cap \Omega_{[t_\zeta, t_{\zeta+1})} \neq \emptyset \\ Q_j \setminus \Phi_j & \text{if } E_{s_j} \cap \Omega_{[t_\zeta, t_{\zeta+1})} = \emptyset \end{cases} \tag{8.9}$$

where $Y_{j_{[t_\zeta, t_{\zeta+1})}} \in \mathbb{R}^{q_j}$ are an arbitrary combination of observations taken from the measurable outputs $(\mathbf{y_k})$ of any system in the network in the interval $[t_\zeta, t_{\zeta+1})$. $E_{s_j} \subseteq E$ is the set of events that can be detected by s_j resulting in the indicator returning a result in $\Phi_j \subseteq Q_j$. The value of indicator s_j is given by:

$$\nu_j = \begin{cases} 1 & g_j(Y_{j_{[t_\zeta, t_{\zeta+1})}}) \in \Phi_j \\ 0 & \text{otherwise} \end{cases} \tag{8.10}$$

Essentially, the j-th indicator function will return a value in a well defined region (Φ_j) of $\mathbb{R}^\kappa$ when one of a set of events (E_{s_j}) that is detectable by s_j occurs in the interval $[t_\zeta, t_{\zeta+1})$.

The generality of this approach can be seen in that a particular indicator function does not need to detect a given event with perfect certainty but rather that the indicator function will determine that one of a set of events has occurred with perfect certainty. This reduces considerably the burden placed on the designer of the indicator functions. The only assumption made here is that it is possible to create an indicator function with the required properties.

There are a number of methods of creating indicator functions and all can be suitable for use within the framework outlined. Existing work has generally focussed on residual indicators detecting a difference between the nominal and actual trajectory of the system as in [25]. Recent work on the structural diagnosis of hybrid networks has shown that such indicators can work by detecting the mode changes directly as in [14].

8.5.1 Event Dependency/Fault Signature Matrix

An alternative simple representation of the events detectable by a given indicator function can be achieved using an event dependency matrix as in [18] or equivalently a fault signature matrix as in [97] . We define this to be:

$$D_{S,E}(j,i) = \begin{cases} 1 & e_i \in E_{s_j} \\ 0 & \text{otherwise} \end{cases} \tag{8.11}$$

Here $D_{S,E}$ is an $m \times n$ matrix showing how events are detectable through indicator functions. The question that arises is what conditions on the matrix $D_{S,E}$ will allow us to guarantee that every event in the network is diagnosable in all time intervals $[t_\zeta, t_{\zeta+1})$, which is addressed in Section 8.6.1.

8.5.2 Example Continued - Fault Signature Matrix

In section 7.5 of chapter 7, the diagnosis method presented is equivalent to having indicator functions that allow the node and link failures to be detected by directly measuring the change in the operational mode. The diagnosis could indicate either that the link itself had failed, or that the node at the end of the link had failed, without being able to distinguish which. Using an appropriate combination of indicators could help distinguish between faults.

If we assume that all possible event indicators can be implemented, then each of the four nodes will have three unique event indicators corresponding to all the link and node failures that can be measured from that node. For example, measurements from node N_1 can trigger indicators s_1, s_2, s_3 (shown in Figure 8.2). A 1 indicates that the indicator (in the row of the matrix) will trigger upon occurrence of the event (and a 0 indicates that the event does not trigger the corresponding indicator). Hence indicator s_1 will trigger either upon occurrence of events N_2 or L_1. Similarly, indicator s_2 will trigger upon occurrence of event N_3 or L_2, and indicator s_3 upon occurrence of events N_4 or L_5.

The corresponding fault signature matrix for all event indicators for the example system is shown in figure 8.2. It is from this set of estimators that we must choose the indicator sets for both fault detection and fault isolation.

Indicator	N_1	N_2	N_3	N_4	L_1	L_2	L_3	L_4	L_5	L_6
s_1	0	1	0	0	1	0	0	0	0	0
s_2	0	0	1	0	0	1	0	0	0	0
s_3	0	0	0	1	0	0	0	0	1	0
s_4	1	0	0	0	1	0	0	0	0	0
s_5	0	0	1	0	0	0	0	0	0	1
s_6	0	0	0	1	0	0	0	1	0	0
s_7	1	0	0	0	0	1	0	0	0	0
s_8	0	1	0	0	0	0	0	0	0	1
s_9	0	0	0	1	0	0	1	0	0	0
s_{10}	1	0	0	0	0	0	0	0	1	0
s_{11}	0	1	0	0	0	0	0	1	0	0
s_{12}	0	0	1	0	0	0	1	0	0	0

Figure 8.2: The fault signature matrix for the complete set of event indicators that are available in the four node hybrid network in this example.

8.6 Diagnosability of Hybrid Dynamical Networks

We have shown, in Lemma 2, that for the hybrid dynamical network to be diagnosable we require the network events to be diagnosable in all time intervals $[t_\zeta, t_{\zeta+1})$. We now present the conditions under which the network events E are diagnosable in each time interval $[t_\zeta, t_{\zeta+1})$ with event indicators S. We present two preliminary definitions on diagnosability that formalize the results we seek.

8.6.1 Diagnosability

We consider two definitions of diagnosability that correspond to the fault detection and fault isolation problem. We refer to them as weak diagnosability and strong diagnosability respectively.

Definition 61 (Weakly Diagnosable)**.** A network is weakly diagnosable with respect to a subset of events ($\bar{E} \subseteq E$) with a given set of event indicators ($\bar{S} \subseteq S$) if the set of event indicators is able to detect the occurrence of events in the set ($\bar{E}$) but cannot differentiate uniquely between them.

Definition 62 (Strongly Diagnosable)**.** A network is strongly diagnosable with respect to a subset of events ($\bar{E} \subseteq E$) with a given set of event indicators ($\bar{S} \subseteq S$) if the set of event indicators is able to detect the occurrence of any event in the set ($\bar{E}$) and distinguish between them, and if the diagnosed set of events (Ω_{diag}) is equal to the true event set (Ω). That is $\Omega = \Omega_{\text{diag}}$.

Remark 8.6.1. The definition of strong diagnosability can be applied to both single faults or multiple faults. In the case of multiple faults there are $2^n - 1$ possible sets of faults corresponding to all possible combinations of faults occurring. Meeting the requirement of strong diagnosability for every possible set of faults can be seen as a very complicated problem. For this reason we

say that a hybrid dynamical network is strongly diagnosable for a given set of faults. If, for example, we were interested in determining the simultaneous occurrence of pairs of events $((e_\alpha, e_\beta)$ for all $\alpha, \beta \in 1, 2, \cdots, n$, such that $\alpha \neq \beta)$ then we would say that the hybrid network is strongly diagnosable for all possible pairs of events.

We now present the conditions for fault detection and fault isolation; the conditions required to satisfy the strong and weak diagnosability conditions.

8.6.2 Weak Diagnosability - Fault Detection

The fault detection problem is to determine if the chosen indicator functions $s_j \in \bar{S}$ will ensure that the occurrence of any event(s) will result in one of the chosen event indicators being active $(v_j = 1)$. Formally this gives us:

Theorem 8.6.2. *Given the set of event indicators $\bar{S}$ and the set of events we wish to detect $(\bar{E})$ then the hybrid dynamical network is weakly diagnosable if:*

$$\bigcup_{\{j|s_j \in \bar{S}\}} E_{s_j} = \bar{E} \tag{8.12}$$

This corresponds to there being a one in all columns of the corresponding fault signature matrix $(D_{\bar{S},\bar{E}})$.

Proof. It can be verified that this condition satisfies the weak diagnosability condition given in definition 61.

Let Ω be the set of active events. For the set of events $\bar{E}$ that can be detected by the set of indicators, at least one indicator will be active if an event from $\bar{E}$ occurs: $\forall s_j \subseteq \bar{S}, \exists v_j = 1$. Let the observation be that at least one indicator from the indicator set $\bar{S}$ is triggered. If we take it that $\Omega \not\subseteq \bar{E}$, then it means the observation will be that no indicators from $\bar{S}$ is active: $\forall e_j \subseteq \bar{S}, v_j = 0$. This is clearly a contradiction to the actual observation obtained. Hence $\Omega \subseteq \bar{E}$. Since the total number of events that can be detected by the indicator set $\bar{S}$ is given by $\bigcup_{\{j|s_j \in \bar{S}\}} E_{s_j}$, we can write $\bigcup_{\{j|s_j \in \bar{S}\}} E_{s_j} = \bar{E}$.

The corresponding proof for the fault signature matrix follows from [18] which first established the result in the context of probing. □

8.6.2.1 Example Continued - Fault Detection

From the complete set of mode estimators given in figure 8.2 it is possible to determine if a given indicator set will satisfy the fault detection condition. Ignoring the method by which this set is chosen we are able to verify easily, using the earlier conditions, that the set in figure 8.4 satisfies the weak diagnosability condition.

For example, if event L_1 occurred, then both indicators s_1 and s_4 will be active. However we cannot distinguish between whether L_1 or N_1 occurred.

Indicator	N_1	N_2	N_3	N_4	L_1	L_2	L_3	L_4	L_5	L_6
s_1	0	1	0	0	1	0	0	0	0	0
s_2	0	0	1	0	0	1	0	0	0	0
s_3	0	0	0	1	0	0	0	0	1	0
s_4	1	0	0	0	1	0	0	0	0	0
s_5	0	0	1	0	0	0	0	0	0	1
s_6	0	0	0	1	0	0	0	1	0	0
s_8	0	1	0	0	0	0	0	0	0	1
s_9	0	0	0	1	0	0	1	0	0	0

Figure 8.3: One possible set of estimators that satisfy the fault detection condition.

Subsequently isolating the location of the fault requires additional conditions to be satisfied and we address this in the following section.

8.6.3 Strong Diagnosability - Fault Isolation

The problem of fault isolation is to determine a set of event indicators such that each fault can be uniquely identified. We will address single fault and multiple fault isolation separately.

8.6.3.1 Single Fault Isolation

The case of single fault isolation, where only a single fault is active at any given time ($|\Omega| = 1$), is the less general fault isolation problem but due to its connection with previous results in [18] it is presented independently. Formally we have:

Theorem 8.6.3. *Given the set of event indicators $\bar{S}$ and the single fault we wish to detect ($e_i \in \bar{E}$).*
Let $\bar{S}_{on}$ be the set of active indicators when e_i occurs: $\bar{S}_{on} \subseteq \bar{S}$ where $\forall j, s_j \in \bar{S}, v_j = 1$.
Let $\bar{S}_{off}$ be the set of inactive indicators when e_i occurs: $\bar{S}_{off} \subseteq \bar{S}$ where $\forall j, s_j \in \bar{S}, v_j = 0$.
If $\forall p, s_p \in \bar{S}_{on}, e_i \in E_{s_p}, \forall e_k \in E_{s_p}$ where $e_k \neq e_i, \exists q, s_q \in \bar{S}_{off}$ where $e_k \in E_{s_q}$, then the hybrid dynamical network will be single fault isolable if it is weakly diagnosable and

$$\Big(\bigcap_{\{j|s_j \in \bar{S}, e_i \in E_{s_j}\}} E_{s_j}\Big) \cap \Big(\bar{E} \setminus \bigcup_{\{j|s_j \in \bar{S}, e_i \notin E_{s_j}\}} E_{s_j}\Big) = \{e_i\} = \Omega \tag{8.13}$$

This corresponds to all the columns of the fault signature matrix ($D_{\bar{S},\bar{E}}$) being unique, with at least a one in every column.

Proof. We are taking the intersection of the possible events as determined by the active event indicators with the possible events as determined by the inactive indicators. We can verify that this condition satisfies the requirements of the strong diagnosability condition given in definition 62. Let $e_i \in \bar{E}$ be the event we would like to detect uniquely. Let the observation be that the set of indicators $\bar{S}_{on} \subseteq \bar{S}$ is triggered.

e_i is weakly diagnosable by the set of indicators $\bar{S}$ if $e_i \in \bar{E}$ (as given in theorem 8.6.2). This implies that there at least one active indicator when e_i occurs: $\forall s_j \subset \bar{S}, \exists v_j = 1$.
Because only one event could have occurred (single fault), e_i is part of the event set for every active indicator. Thus we can write $\forall s_j \subseteq \bar{S}, \forall v_j = 1, e_i \in \bigcap_{\{j|s_j \in \bar{S}, e_i \in E_{s_j}\}} E_{s_j}$.
This forcibly implies that e_i is not in the inactive set of indicators: $e_i \notin \bigcup_{\{j|s_j \in \bar{S}, e_i \notin E_{s_j}\}} E_{s_j}$ and thus $e_i \in (\bar{E} \setminus \bigcup_{\{j|s_j \in \bar{S}, e_i \notin E_{s_j}\}} E_{s_j})$.
Let e_k be an event in the inactive set of indicators: $e_k \in E_{\bar{S}_{on}}$. We can also write $e_k \in \bigcup_{\{j|s_j \in \bar{S}, e_i \notin E_{s_j}\}} E_{s_j}$.
Also, let e_k be in the set of active indicators: $e_k \in E_{\bar{S}_{off}}$. We can also write $e_k \in \bigcap_{\{j|s_j \in \bar{S}, e_k \in E_{s_j}\}} E_{s_j}$.
Since $e_k \in E_{\bar{S}_{off}}$ and $e_i \notin E_{\bar{S}_{off}}$, then $e_k \neq e_i$. This allows us to use the inactive indicator set to eliminate events that appear in both the active and inactive indicator sets as they could not have occurred based on the observation.

Therefore, e_i can be found from the intersection of possible events given by set of active indicators and the possible events determined by the inactive indicators and we can write $\{e_1\} = (\bigcap_{\{j|s_j \in \bar{S}, e_i \in E_{s_j}\}} E_{s_j}) \cap (\bar{E} \setminus \bigcup_{\{j|s_j \in \bar{S}, e_i \notin E_{s_j}\}} E_{s_j})$.

The corresponding proof for the fault signature matrix follows from [18] which first established the result in the context of probing. □

8.6.3.2 Example Continued - Single Fault Isolation

From the complete set of mode estimators given in figure 8.2, it is possible to determine if a given indicator set will satisfy the single fault isolation condition. Ignoring the method by which this set is chosen we are able to verify easily, using the earlier conditions, that the set in figure 8.4 allows single fault isolation to occur.

For example, if N_1 occurred, the active indicator set is $\{s_4, s_7\}$. The corresponding events that can be detected are $\{N_1, L_1, L_2\}$. If L_1 had occurred, then indicator s_1 should have been on. Hence L_1 did not occur. Similarly, if L_2 had occurred, the indicator s_2 should have been on. Hence L_2 did not occur. Hence N_1 must be the event that occurred.

We show next the conditions for which multiple faults can be isolated presenting the most general case of the fault isolation condition.

8.6.3.3 Multiple Fault Isolation

When multiple faults occur (that is $|\Omega| \geq 1$) it is no longer possible to isolate the faults using the fault isolation condition defined previously. We now seek to isolate the occurrence of multiple faults by determining the events that are not occurring through exploiting our knowledge of the indicators that are not active. Formally this gives us:

Theorem 8.6.4. *Given the set of event indicators $\bar{S}$ and the set of faults we wish to detect ($E_{faults} \subseteq \bar{E}$). Let $\bar{S}_{on}$ be the set of active indicators when events from E_{faults} occur: $\bar{S}_{on} \subseteq \bar{S}$ where $\forall j, s_j \in \bar{S}, v_j = 1$.*

Indicator	N_1	N_2	N_3	N_4	L_1	L_2	L_3	L_4	L_5	L_6
s_1	0	1	0	0	1	0	0	0	0	0
s_2	0	0	1	0	0	1	0	0	0	0
s_3	0	0	0	1	0	0	0	0	1	0
s_4	1	0	0	0	1	0	0	0	0	0
s_5	0	0	1	0	0	0	0	0	0	1
s_6	0	0	0	1	0	0	0	1	0	0
s_7	1	0	0	0	0	1	0	0	0	0
s_8	0	1	0	0	0	0	0	0	0	1
s_9	0	0	0	1	0	0	1	0	0	0

Figure 8.4: One possible set of estimators that satisfy both the fault detection and single fault isolation conditions presented earlier.

Let $\bar{S}_{off}$ be the set of inactive indicators when events from E_{faults} occur: $\bar{S}_{off} \subseteq \bar{S}$ where $\forall j, s_j \in \bar{S}, v_j = 0$.
Let $E_{faults} = \{e_{f_1}, \cdots, e_{f_k}\}$. If $\forall p, s_p \in \bar{S}_{on}, \forall i, i \in \{f_1, \cdots, f_k\}, e_i \in E_{s_p}, \forall e_k \in E_{s_p}$ where $e_k \neq e_i, \exists q, s_q \in \bar{S}_{off}$ where $e_k \in E_{s_q}$, then the hybrid dynamical network is multiple fault isolable for E_{faults} if it is weakly diagnosable and

$$\bar{E} \setminus \Big(\bigcup_{\{j | s_j \in \bar{S}, E_{faults} \cap E_{s_j} = \emptyset\}} E_{s_j} \Big) = E_{faults} = \Omega \tag{8.14}$$

This corresponds to a one in every column of the fault signature matrix ($D_{\bar{S},\bar{E}}$) to satisfy weak diagnosability as well as a one in every column of the submatrix ($D_{\underline{S},\bar{E}\setminus E_{faults}} \subseteq D_{\bar{S},\bar{E}}$) where $\underline{S} = \{s_j | E_{faults} \cap E_{s_j} = \emptyset\}$ is the set of event indicators not sensitive to any of the fault events.

Proof. Again it is easy to verify that the condition presented satisfies the strong diagnosability condition of definition 62.

Let $E_{\text{faults}} = \{e_{f_1}, \cdots, e_{f_k}\}$ be the set of events we would like to detect uniquely. Let the observation be that the set of indicators $\bar{S}_{on} \subseteq \bar{S}$ is triggered.
E_{faults} is weakly diagnosable by the set of indicators S if $\forall i, i \in \{f_1, \cdots, f_k\}, e_i \in E$ (as given in theorem 8.6.2). Let e_k be an event in the inactive set of indicators: $e_k \in E_{\bar{S}_{on}}$. We can also write $e_k \in \bigcup_{\{j | s_j \in \bar{S}, E_{\text{faults}} \cap E_{s_j} = \emptyset\}} E_{s_j}$.
Also, let e_k be in the set of active indicators: $e_k \in E_{\bar{S}_{off}}$. We can also write $e_k \in \bigcup_{\{j | s_j \in \bar{S}, E_{\text{faults}} \in E_{s_j}\}} E_{s_j}$.
Since $e_k \in E_{\bar{S}_{off}}$ and $\forall i, i \in \{f_1, \cdots, f_k\}, e_i \notin E_{\bar{S}_{off}}$, then $e_k \neq e_i$. This allows us to use the inactive indicator set to eliminate events that appear in both the active and inactive indicator sets as they could not have occured based on the observation.
Hence for each active indicator, only the events that do not appear in any inactive indicator could have happened. They can thus be identified uniquely and we can write $E_{\text{faults}} = \bar{E} \setminus (\bigcup_{\{j | s_j \in \bar{S}, E_{\text{faults}} \cap E_{s_j} = \emptyset\}} E_{s_j})$.

Indicator	N_1	N_2	N_3	N_4	L_1	L_2	L_3	L_4	L_5	L_6
s_1	0	1	0	0	1	0	0	0	0	0
s_3	0	0	0	1	0	0	0	0	1	0
s_7	1	0	0	0	0	1	0	0	0	0
s_8	0	1	0	0	0	0	0	0	0	1
s_{10}	1	0	0	0	0	0	0	0	1	0
s_{11}	0	1	0	0	0	0	0	1	0	0
s_{12}	0	0	1	0	0	0	1	0	0	0

Figure 8.5: The event indicators necessary to allow multiple fault diagnosability for node N_4 and link L_1 failures.

The corresponding condition for the fault signature matrix can be seen as simply the fault detection condition for those events that are not occurring. □

The strong diagnosability condition for multiple faults is also applicable for single faults and is thus the most general diagnosability result presented in this work.

8.6.4 Example Continued - Multiple Fault Isolation

From the complete set of mode estimators given in figure 8.2 it is possible to determine if a given indicator set will satisfy the multiple fault isolation condition for a given fault set or class of fault sets. For clarity we focus on the isolation of the pair of faults when node N_4 and link L_1 fail. We are able to verify easily, using the earlier conditions, that the set in figure 8.5 satisfies the conditions for the multiple faults indicated.

When N_4 and L_1 occur, the active indicator set is $\{s_1, s_3\}$. The corresponding events that can be detected are $\{N_2, L_1, N_4, L_5\}$. If N_2 had occurred, then indicator s_8 should have been on. Hence N_2 did not occur. Similarly, if L_5 had occurred, the indicator s_{10} should have been on. Hence events N_2 and L_5 did not occur. Hence N_4 and $|_1$ must be the events that occurred.

It should be noted that this set of indicators will not necessarily satisfy the multiple fault diagnosis condition for an arbitrary set of faults, highlighting the complexity of the arbitrary multiple fault isolation problem.

8.6.5 Complexity

In all cases in the preceding analysis it should be noted that the complexity of the algorithms is polynomial as it consists of checking a simple set theoretic or matrix condition.

8.7 Determining the Minimal Indicator Set for Diagnosability

We have thus far presented conditions that guarantee the diagnosability of a given hybrid dynamical network. It remains to show how to determine the minimal set of event indicators $S_{\min} \subseteq S$

such that the diagnosability conditions are satisfied. We will present a complexity analysis of each of these problems to guide the development of algorithms for finding such a minimum diagnosability set.

8.7.1 Complexity Analysis

Theorem 8.7.1 (Complexity of Indicator Selection). *Indicator selection for fault detection is NP-hard*

Proof. Fault detection is exactly the minimum set cover problem [60] which is known to be NP-hard. □

Theorem 8.7.2 (Single Fault Isolation Complexity). *Single fault isolation is NP-hard.*

Proof. This was shown to be true in [18] using a reduction from the fault detection problem outlined previously. □

Theorem 8.7.3 (Multiple Fault Isolation Complexity). *Multiple fault isolation is NP-hard.*

Proof. Multiple fault isolation is the generalisation of single fault detection. Hence it is also NP-hard. □

Having shown the computational difficulty of each of these problems we see that finding the minimal set that guarantees diagnosability is a complicated task. The reader is directed to results in [18] and [30] for suitable polynomial time methodologies giving algorithms that would allow the near optimal minimum diagnosability set to be determined. Additional methods of finding such minimal diagnosability sets is an active area of future work.

8.8 Conclusion

In this chapter we have presented a general framework for determining the diagnosability of hybrid dynamical networks using a general class of indicator functions. We show that determining the operational mode of each system in the network reduces to determining the occurrence of all events in the network. Weak and strong diagnosability results are hence presented in this context. The generality of the approach provides flexibility in designing appropriate indicator functions to exploit this result.

We have shown that determining if a given set of indicators satisfies the diagnosability conditions has polynomial complexity however choosing a minimum set of such indicators that still guarantees diagnosability is NP-hard. This suggests that future work in this area should focus on the development of algorithms that can recover a near optimal solution in polynomial time. These results would be directly applicable to the construction of minimum order diagnosis engines for a variety of real world hybrid dynamical networks.

Chapter 9

Conclusion

9.1 Evaluation

Distributed systems (*i.e.* systems consisting of many interconnected components) are an important part of everyday life. From distributing electricity to travelling across the world, we have come to take their operations for granted. Failures on such systems as electricity networks or aircrafts would cause major inconvenience and can be dangerous. Hence, it is really important to know what is happening on them in order to take remedial actions whenever necessary and prevent major failures. Hence the diagnosis of distributed systems has become increasingly important as those systems become more pervasive and increase in complexity and size.

The main objective of this thesis was to contribute methods to the diagnosis of distributed systems that exhibit an emergent complex behaviour. Research has been done from two key perspectives, a discrete event system (DES) viewpoint and a dynamical system viewpoint, which were unified into a hybrid system approach. Methodologies from the DX community (with roots in Computer Science) and the FDI community (with roots in Control Theory) have been extended and integrated. A unifying approach provides a powerful framework for handling systems with complex dynamics.

The attraction of a DES viewpoint resides in the possibility of using logic-based methodologies to reason on system behaviour. However, because a distributed system consists of many interconnected components, obtaining a global diagnosis by synchronising the diagnoses on each component quickly becomes unmanageable as the number of components considered increases. If each component is modeled as an automaton, which is the case in my work, then the complexity of doing this global computation is exponential to the number of components in the system. To circumvent this problem, I have proposed a distributed approach for diagnosis which is based on transforming the topological graph of the global system into a junction tree. Each cluster of the junction tree is a sub-system of the network. The properties of a junction tree make it an elegant method for managing the complexity of reasoning on the entire system because local consistency among the clusters of the tree is sufficient to ensure global consistency. Diagnosis using junction

trees was explored on systems of different topological shapes and sizes and results confirmed the intuition that the methodology works best for systems that have a tree or near-tree structure. It was found that the proposed diagnosis methodology works well on small world network models with a tree-like structure (shown to be good models for electricity networks [98]), thus validating its use for our motivating application.

For systems where the components are highly connected among themselves, the proposed diagnosis method would still have a high cost. A way to handle this issue was proposed where connections were selectively ignored on the system when building the junction tree. Removing connections would normally decrease the accuracy of diagnosis but an apriori accuracy analysis is performed to determine which connections can be safely ignored. Hence it was possible to consider sub-systems of an interconnected system for diagnosis of the latter without compromising the accuracy of the diagnosis obtained. However, this introduced a trade-off in time as more time would then be required for obtaining a diagnosis result.

While a DES approach works well for reasoning on an abstracted discrete event model of a physical system, it is not suited for capturing the dynamical evolution of the system. Information contained in the dynamics of the system is useful for diagnosis at the behavioural level. Hence, I combined both a DES approach and a dynamical system approach into a hybrid framework to obtain an augmented diagnosis approach. A DES model was extracted from the dynamical operations of a system by detecting changes in its fundamental governing dynamics, and not by residual estimation on its output trajectory. This technique has the advantage of being applicable to systems that might not be well characterised and where parameter drift is present. The proposed hybrid approach was deployed on a small network example and showed its capability for diagnosis in a situation where a solely dynamical approach failed.

In a large complex network, the task of determining whether the occurrence of all events in the network can be diagnosed, *i.e.* determining the diagnosability of the network, is significantly complicated due to the interaction of the events in the many interconnected components. Events occurring in one component of the network often affect other components as well, thereby generating simultaneous or sequential sets of events. I worked within a hybrid system framework to define indicator functions that detect the onset of events in the system from its continuous operation. They thus acted as an interface between the discrete operation and continuous operation of the system. These indicator functions were then used to derive diagnosability conditions for the detection and isolation of multiple, arbitrary sequential or simultaneous events in the network. Diagnosability conditions are dependent on the number of events in the system rather than on the number of interconnected components. This insight is interesting because the interconnection of many dynamical systems is known to lead to complex emergent behaviour not observed in the individual systems. Thus, having a way to characterise hybrid dynamical networks in terms of their diagnosability using event indicators, provides a tool for determining how best to diagnose and control such networks and render them more fault-tolerant. While hybrid dynamical networks were the focus, since the diagnosability results were defined on the discrete part of the system,

they can be extended to any DES model.

9.2 Future Work

The distributed diagnosis method presented in chapter 4 relies on a static construction of a junction tree based on the topology of the system under consideration, performed apriori. An immediate area of future work would be the investigation of a dynamic approach to the construction of the junction tree used for diagnosis. The diagnosis algorithm could be started while the junction tree is being computed and the intermediate diagnostic result could be used to inform the construction of the junction tree dynamically. The intuition is that some connections in the system topology can be removed when no communication happens through them, which would simplify the topological graph and hence reduce the cost of diagnosis. However, this is not trivial as the construction of junction trees must satisfy the junction tree properties at all times. An interesting investigation would be to try other measures of cost than the tree-width. A possible candidate would be length of longest branch (from root to leaf), or a combination of tree-width and length of branch. Another interesting consideration is possible privacy issues on the network. It might be desirable to enforce rules about communication between components. In that case, diagnosis is complicated because an agent (*i.e.* a component) needs to choose which other agents it can communicate with, thereby adding a constraint to the construction of the junction tree for diagnosis. These considerations would make the diagnosis problem more challenging and would help refine methodologies for application to real-world systems.

In chapter 5, a method was presented for performing an off-line accuracy analysis in order to consider which connections can be ignored when doing diagnosis. This allowed smaller sub-systems to be considered thus reducing the complexity of the diagnosis. However, in doing so, there is a trade-off involved as the time taken to gather enough observations to return a useful diagnosis result is longer. The fairness assumption made implies that the delay required for a sub-system to become accurate, *i.e.* return the same diagnostic result as the overall system, is expected to be reasonable. However, there can be situations where a better bound on this delay is required, *e.g.* where the diagnosis result is needed in a set timeframe in order to take action in critical situations. An interesting area of future work is to investigate including a bounded delay in the definition of accuracy. To obtain a smaller delay, bigger sub-systems need to be considered. The trade-off between sub-system size and delay would need to be explored. We could further improve the accuracy criterion by considering cost in the evaluation of the trade-off required. A future research direction would be to use those three factors (sub-system size, delay and cost) to come up with an accuracy function that returns a real value instead of a boolean result. This quantification could be used to evaluate and compare all possible sub-systems of the system by exploration on a lattice. The notion of cost of diagnosis could be further refined by considering other measures than tree-width. Possibilities include length of longest branch, total number of clusters in the junction tree or proportion of observable events. The accuracy criterion has been defined with respect to a

single fault only. A further improvement would be to consider multiple faults.

In the proposed hybrid system framework, the interface between continuous and discrete dynamics was handled by performing an abstraction based on changes in the fundamental governing dynamics of the system. While this has advantages, as discussed previously, an improved methodology would look at integrating this approach with one using residual generation techniques to harness the advantages of both. Residual methods detect changes by looking at deviations from the output trajectory of the system. They require apriori knowledge of system operations and parameter values, and no drift in parameter values. During the operation of the system, there will be large chunks of time for which these conditions hold. During such periods, it would be more fitting to use residual estimation since it is the less costly method. Another option is to apply both approaches at the same time to build in a higher safety factor for diagnosis on the system, relevant to safety critical systems. A further research direction would be to enhance diagnosis by using machine learning techniques for situations where little apriori knowledge is available. This would be the case when non-linear components are present in the system as they often have unknown interactions (*e.g.* renewable energy sources such as solar or wind on a smart grid). Finally, the author hopes that the work presented in this thesis contributes to the application of manageable diagnosis methodologies to real world large scale problems and thus makes a positive impact to society at large.

Bibliography

[1] T. Abdelzaher, Y. Diao, J.L. Hellerstein, S. Singhal C. Lu, and X. Zhu. Recent advances in the application of control theory to network and service management. In *11th IFIP/IEEE International Symposium on Integrated Network Management Tutorial*, New York, USA, 2009.

[2] G.W. Alexander, S.L. Corbin, and W.J. McNutt. Influence of design and operating practices on excitation of generator step-up transformers. *IEEE Transactions on Power Apparatus and Systems*, PAS-85(8):901 – 909, 1966.

[3] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[4] B.D.O. Anderson and J.B. Moore. *Optimal filtering / Brian D. O. Anderson, John B. Moore.* Prentice-Hall, Englewood Cliffs, N.J. :, 1979.

[5] B.D.O Anderson and J.B. Moore. *Optimal Filtering.* Dover Publications, Inc, 2005.

[6] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[7] BillC at en.wikipedia. Transformer, wikipedia, 2006.

[8] MBizon at wikipedia. Electricity grid, wikipedia, 2010.

[9] B. el Ayeb, P. Marquis, and M. Rusinowitch. Deductive/abductive diagnosis: The da-principles. In *ECAI*, pages 47–52, 1990.

[10] V.K Balakrishnan. *Graph Theory.* McGraw-Hill, 1997.

[11] C. Bartoletti, M. Desiderio, D. Di Carlo, G. Fazio, F. Muzi, G. Sacerdoti, and F. Salvatori. Vibro-acoustic techniques to diagnose power transformers. *IEEE Transactions on Power Delivery*, 19(1):221 – 229, 2004.

[12] M. Bayoudh, L. Travé-Massuyès, and X. Olive. Towards active diagnosis of hybrid systems. In *International Workshop on Principles of Diagnosis*, Blue Mountains, Australia, 2008.

[13] L. Blackhall, P. Kan John, A. Grastien, and D. Hill. Diagnosability of hybrid dynamical networks using indicator functions [extended version]. In *20th Workshop on the Principles of Diagnosis(DX-09)*, Stockholm, Sweden, 2009.

[14] L. Blackhall and P. Kan John. Model-based diagnosis of hybrid dynamical networks for fault tolerant control. In *19th International Workshop on Principles of Diagnosis*, Blue Mountains, NSW, Australia, 2008.

[15] L. Blackhall and M. Rotkowitz. Recursive sparse estimation using a gaussian sum filter. In *IFAC 2008*, 2008.

[16] L. Blackhall and M. Rotkowitz. Maximum aposteriori vs maximum probability recursive sparse estimation. In *Proceedings of the European Control Conference - To Appear*, 2009.

[17] T. Boukhobza, F. Hamelin, and S. Martinez-Martinez. State and input observability for structured linear systems: A graph-theoretic approach. *Automatica*, 43:1204–1210, 2007.

[18] M. Brodie, I. Rish, S. Ma, A. Beygelzimer, and N. Odintsova. Strategies for problem determination using probing. Technical report, IBM T.J. Watson Research Center, 2002.

[19] T. Bylander. Some causal models are deeper than others. *Artificial Intelligence in Medicine*, 2(3):123 – 128, 1990.

[20] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[21] L. Chang and A.K. Mackworth. Constraint-based inference using local consistency in junction graphs. In *Proceedings of the ECAI 2006 Workshop on Inference Methods Based on Graphical Structures of Knowledge, WIGSK 2006*, pages 1–6, Riva del Garda, Italy, August 2006.

[22] G.R Chen. Complex networks: Modeling, dynamics and control. Lecture notes for EE6605, 2009.

[23] Zhe Chen. Bayesian filtering: from Kalman filters to particle filters, and beyond. Technical report, McMaster University, 2003.

[24] S. Chib. Estimation and comparison of multiple change-point models. *Journal of Econometrics*, 86(2):221–241, June 1998.

[25] V. Cocquempot, T. El Mezyani, and M. Staroswieckit. Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference*, 2004.

[26] L. Console, D. Theseider Dupré, and P. Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.

[27] L. Console and P. Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.

[28] M.-O. Cordier, P. Dague, M. Dumas, F. Levy, J. Montmain, M. Staroswiecki, and L. ravé Massuyès. AI and automatic control approaches of model-based diagnosis: Links and underlying hypotheses. In A.M. Edelmayer, editor, *4th IFAC Symosium on Fault Detection Supervision and Safety for Technical Processes*, volume 1, pages 274 – 279, 2000.

[29] M.-O. Cordier and A. Grastien. Exploiting independence in a decentralised and incremental approach of diagnosis. In M. Veloso, editor, *Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 292–297. AAAI press, 2007.

[30] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2 edition, 2001.

[31] J.H Dann and J.J Dann. The people's physics book, 2006.

[32] Adnan Darwiche. Model-based diagnosis under real-world constraints. *AI Magazine, Summer*, 21:57–73, 2000.

[33] O. Dressler, C. Böttcher, M. Montag, and A. Brinkop. Qualitative and quantitative models in a model-based diagnosis system for ballast tank systems. In *Proceedings of the International Conference on Fault Diagnosis (TOOLDIAG-93)*, Toulouse, France, 1993.

[34] P. Erdós and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290 – 297, 1959.

[35] P. Erdós and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17 – 61, 1960.

[36] E. Fabre, A. Benveniste, and Cl. Jard. Distributed diagnosis for large discrete event dynamic systems. In *Proceedings of the IFAC World Congress*, 2002.

[37] Online Study Site for Electrical Engineering. Definition of transformer, 2013.

[38] S. Genc and S. Lafortune. Distributed diagnosis of place-bordered Petri nets. *IEEE Transactions on Automation Science and Engineering*, 4(2):206–219, April 2007.

[39] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In R. Holte, editor, *Nineteenth National Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[40] A. Grastien, P. Haslum, and S. Thiébaux. Conflict-based diagnosis of discrete event systems: Theory and practice. In Gerhard Brewka, Thomas Eiter, and Sheila A. McIlraith, editors, *KR*. AAAI Press, 2012.

[41] P.J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711 – 732, 1995.

[42] IEEE Smart Grid. Smart grid conceptual model, 2012.

[43] M. Hack. Petri net language. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1976.

[44] F. Hamelin, D. Sauter, and D. Theilliol. Some extensions to the parity space method for FDI using alternated projection subspaces. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 3, pages 2407–2412 vol.3, dec 1995.

[45] W. Hamscher, L. Console, and J. de Kleer. *Readings in model-based diagnosis*. Morgan Kaufmann, 1992.

[46] T.A. Henzinger. The theory of hybrid automata. In *Logic in Computer Science, 1996. LICS '96. Proceedings., Eleventh Annual IEEE Symposium on*, pages 278–292, 1996.

[47] M. Hofbaur and B.C Williams. Mode estimation of probabilistic hybrid systems. In *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control*, pages 253–266, 2002.

[48] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*. Pearson Addison-Wesley, Upper Saddle River, NJ, 3. edition, 2007.

[49] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[50] R. Isermann. Supervision, fault-detection and fault-diagnosis methods – an introduction. *Control Engineering Practice*, 5(5):639–652, 1997.

[51] J. de Kleer, A.K. Mackworth, and R. Reiter. *Characterizing diagnoses and systems*, pages 54–65. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[52] J. de Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[53] F.V. Jensen and F. Jensen. Optimal junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, Washington*, 1994.

[54] J. Lunze JM. Blanke, M. Kinnaert and M. Staroswiecki. *Diagnosis and fault-tolerant control*. Springer-Verlag, Berlin, Germany, 2003.

[55] P. Kan John, L. Blackhall, A. Grastien, and D. Hill. Diagnosing structural changes in hybrid dynamical systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS-09)*, Barcelona, Spain, 2009.

[56] P. Kan John, A. Grastien, and Y. Pencolé. Synthesis of a distributed and accurate diagnoser. In *21th Workshop on the Principles of Diagnosis(DX-10)*, Portland, USA, 2010.

[57] P. Kan John, A. Grastien, Y. Pencolé, and P. Ribot. Synthèse d'un diagnostiqueur distribué et précis. In *Reconnaissance des Formes et Intelligence Artificielle, Actes du 17ème congrès francophone AFRIF-AFIA (RFIA-10)*, Caen, France, 2010.

[58] R.E Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[59] P. Kan John and A. Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *European Conference on Artificial Intelligence (ECAI-08)*, Patras, Greece, 2008.

[60] R.M. Karp. *Complexity of computer computations*. Plenum Press, 1972.

[61] H.K. Khalil. *Nonlinear systems*. Prentice Hall, 3rd edition, 2002.

[62] U. Kjrulff. Triangulation of graphs - algorithms giving small total state space, 1990.

[63] Johan De Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[64] X. Koutsoukos, J. Kurien, and F. Zhao. Monitoring and diagnosis of hybrid systems using particle filtering methods. In *In Proceedings of the Fifteenth International Symposium on the Mathematical Theory of Networks and Systems (MTNS 02), University of Notre Dame, South*, 2002.

[65] A. Grastien L. Blackhall, P. Kan John and D.J. Hill. Diagnosability of networks of hybrid systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS-09)*, Barcelona, Spain, 2009.

[66] S. Lai, D. Nessi, M.P. Cabasino, A. Giua, and C. Seatzu. A comparison between two diagnostic tools based on automata and petri nets. In *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES-08)*.

[67] G. Lamperti and M. Zanella. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.

[68] X. Li, X. Wang, and G. Chen. Pinning a complex dynamical network to its equilibrium. *IEEE Transactions on Circuits and Systems-I:Regular Papers*, 51(10):2074–2087, October 2004.

[69] L. Ljung. *System Identification - theory for the user*. PTR Prentice Hall, Upper Saddle River, N.J., 1999.

[70] J. Lunze. Discrete-event modelling and diagnosis of quantized dynamical systems. In *Proceedings of the Tenth International Workshop on Principles of Diagnosis (DX-99)*, pages 147–154, 1999.

[71] J. Lunze and F. Schiller. Fault diagnosis based on a predicate logic description of dynamical systems. In R. Patton, P.M. Frank, and R.N. Clark, editors, *Issues of fault diagnosis for dynamic systems*, pages 485–514. Springer-Verlag, 2000.

[72] S. McIlraith, G. Biswas, D. Clancy, and V. Gupta. Hybrid systems diagnosis. In *Third International Workshop on hybrid systems*, pages 282–295, 2000.

[73] S. McIlraith, G. Biswas, D. Clancy, and V. Gupta. Hybrid systems diagnosis. In *International Workshop On Hybrid Systems, Computation and Control*, Pittsburgh, PA, 23-25 March 2000.

[74] W.J. McNutt. Operation of power transformers during major power system disturbances. Technical report, Power Transformer Department, General Electric, http://store.gedigitalenergy.com/FAQ/Documents/T35/GET-3237.pdf, 1966.

[75] T. Melliti and P. Dague. Generalizing diagnosability definition and checking for open systems: a game structure approach. In *Proceedings of the 21st International Workshop on Principles of Diagnosis (DX-10)*, pages 103–110, 2010.

[76] Anco Muscholl. Lecture notes on Petri nets. `http://www.labri.fr/perso/anca/FDS/Pn-ESTII.pdf`.

[77] S. Narasimhan, F. Zhao, G. Biswas, and E. Hung. Fault isolation in hybrid systems combining model based diagnosis and signal processing. In *Third International Workshop on hybrid systems*, 2000.

[78] M.E.J. Newman and D.J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, December 1999.

[79] O. Noran. The evolution of expert systems. Technical report, School of Computing and Information Technology, Griffith University, http://cit.gu.edu.au/ noran, 2003.

[80] Princeton Course notes from Wikipedia Documents. Electromotive force, 2013.

[81] H.R.B. Orlande, M.J. Colaço, G.S. Dulikravich, F.L.V. Vianna, W.B. da Silva, and H.M. da Fonseca and. Kalman and particle filters - Tutorial 10. `http://www.sft.asso.fr/Local/sft/dir/user-3775/documents/actes/Metti5_School/Lectures&Tutorials-Texts/Text-T10-Orlande.pdf`.

[82] R.J. Patton and J. Chen. Observer-based fault detection and isolation: Robustness and applications. *Control Engineering Practice*, 5(5):671 – 682, 1997.

[83] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence (AIJ)*, 164:121–170, 2005.

[84] X. Pucel. *A unified point of view on diagnosability*. PhD thesis, Université de Toulouse, 1997.

[85] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence Journal (AIJ)*, 32(1):57–95, 1987.

[86] M.I Ribeiro. Kalman and extended Kalman filters: concept, derivation and properties. Technical report, Institute for Systems and Robotics, Portugal, 2004.

[87] P. Ribot, Y. Pencolé, and M. Combacau. Design requirements for the diagnosability of distributed discrete event systems. In *19th International Workshop on Principles of Diagnosis*, Blue Mountains, Australia, 2008.

[88] J. Rintanen. Diagnosers and diagnosability of succint transition systems. In M. Veloso, editor, *Proceedings of the 20th Joint Conference on Artificial Intelligence (AAAI-07)*. AAAI Press, 2007.

[89] M. Sachenbacher and P. Struss. Task-dependent qualitative domain abstraction. *Artificial Intelligence (AIJ)*, 162(1–2):121–143, 2005.

[90] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[91] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.

[92] A. Schumann and J. Huang. A scalable jointree algorithm for diagnosability. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, 2008.

[93] A. Schumann, Y. Pencolé, and S. Thiébaux. Symbolic models for diagnosing discrete-event systems. In *Sixteenth European Conference on Artificial Intelligence (ECAI'04)*, 2004.

[94] K. Sehjong and D.J. Hill. Synchronization of a complex network with switched coupling. In *IFAC 2008*, 2008.

[95] P. Struss. Fundamentals of model-based diagnosis of dynamic systems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997.

[96] R. Su and W. M. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *Transactions on Automatic Control*, 50(12):1923–1935, 2005.

[97] L. Travé-Massuyès, T. Escobet, S. Spanache, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 2004.

[98] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393:440 – 442, June 1998.

[99] B.C Williams, M. Hofbaur, and T. Jones. Mode estimation of probabilistic hybrid systems. Technical Report 6-01, Massachusetts Institute of Technology, May 2001.

[100] G. Yong. Phase transition of tractability in constraint satisfaction and bayesian network inference. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 265–271. Morgan Kaufmann Publishers Inc., 2003.

[101] D. Yu and U. Parlitz. Estimating parameters by autosynchronization with dynamics restrictions. *Physical Review E*, 77(6), 2008.

Appendix A

Examples of networks with small-world topologies

This appendix shows diagrams of 3 large networks with small-world properties for visualisation.

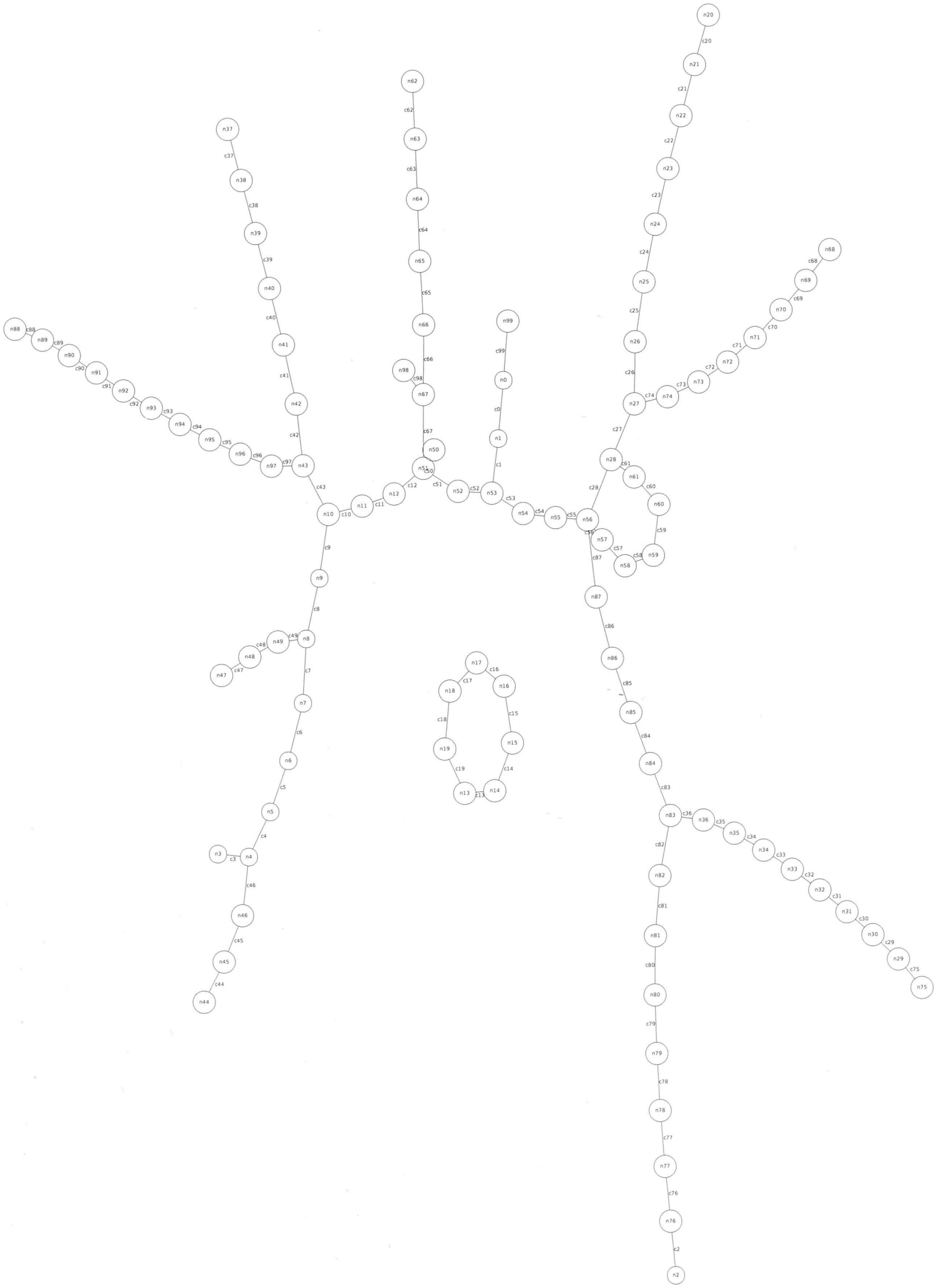


Figure A.1: Small-world network generated from original ring network with parameters: n=100, k=1, p=0.2

Figure A.2: Small-world network generated from original ring network with parameters: n=200, k=1, p=0.1

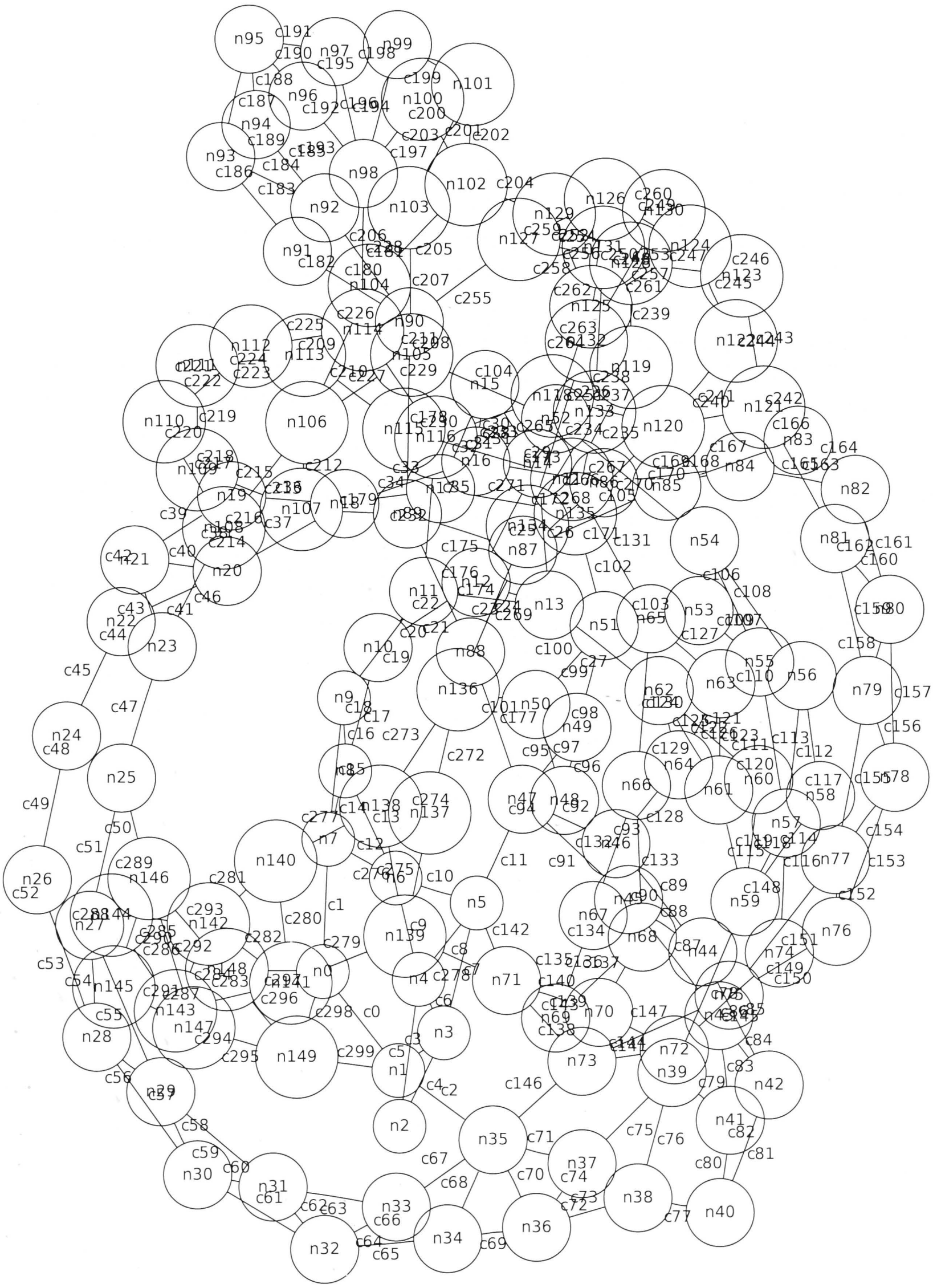


Figure A.3: Small-world network generated from original ring network with parameters: n=150, k=2, p=0.1

Appendix B

Publications Arising from this Thesis

Please note that this list only details those papers that have been accepted for publication, not those papers that have been submitted or that are currently in preparation.

- P. Kan John and A. Grastien. Local consistency and junction tree for diagnosis of discrete-event systems. In *European Conference on Artificial Intelligence (ECAI-08)*, Patras, Greece, 2008

- L. Blackhall and P. Kan John. Model-based diagnosis of hybrid dynamical networks for fault tolerant control. In *19th International Workshop on Principles of Diagnosis*, Blue Mountains, NSW, Australia, 2008

- A. Grastien L. Blackhall, P. Kan John and D.J. Hill. Diagnosability of networks of hybrid systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS-09)*, Barcelona, Spain, 2009

- P. Kan John, L. Blackhall, A. Grastien, and D. Hill. Diagnosing structural changes in hybrid dynamical systems. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS-09)*, Barcelona, Spain, 2009

- L. Blackhall, P. Kan John, A. Grastien, and D. Hill. Diagnosability of hybrid dynamical networks using indicator functions [extended version]. In *20th Workshop on the Principles of Diagnosis(DX-09)*, Stockholm, Sweden, 2009

- P. Kan John, A. Grastien, Y. Pencolé, and P. Ribot. Synthèse d'un diagnostiqueur distribué et précis. In *Reconnaissance des Formes et Intelligence Artificielle, Actes du 17ème congrès francophone AFRIF-AFIA (RFIA-10)*, Caen, France, 2010

- P. Kan John, A. Grastien, and Y. Pencolé. Synthesis of a distributed and accurate diagnoser. In *21th Workshop on the Principles of Diagnosis(DX-10)*, Portland, USA, 2010